AMD

AMD64 Technology

AMD64 Architecture Programmer's Manual Volume 4: 128-Bit Media Instructions

Publication No.	Revision	Date
26568	**3.07**	**December 2005**

Advanced Micro Devices

Contents

Figures

Tables

Revision History

Date	Revision	Description
December 2005	3.07	Made minor editorial and formatting changes.
January 2005	3.06	Added documentation on SSE3 instructions. Corrected numerous minor factual errors and typos.
September 2003	3.05	Made numerous small factual corrections.
April 2003	3.04	Made minor corrections.

Preface

About This Book

This book is part of a multivolume work entitled the *AMD64 Architecture Programmer's Manual*. This table lists each volume and its order number.

Title	Order No.
Volume 1, *Application Programming*	24592
Volume 2, *System Programming*	24593
Volume 3, *General-Purpose and System Instructions*	24594
Volume 4, *128-Bit Media Instructions*	26568
Volume 5, *64-Bit Media and x87 Floating-Point Instructions*	26569

Audience

This volume (Volume 4) is intended for all programmers writing application or system software for processors that implement the AMD64 architecture.

Contact Information

To submit questions or comments concerning this document, contact our technical documentation staff at AMD64.Feedback@amd.com.

Organization

Volumes 3, 4, and 5 describe the AMD64 architecture's instruction set in detail. Together, they cover each instruction's mnemonic syntax, opcodes, functions, affected flags, and possible exceptions.

The AMD64 instruction set is divided into five subsets:

- General-purpose instructions
- System instructions
- 128-bit media instructions

- 64-bit media instructions
- x87 floating-point instructions

Several instructions belong to—and are described identically in—multiple instruction subsets.

This volume describes the 128-bit media instructions. The index at the end cross-references topics within this volume. For other topics relating to the AMD64 architecture, and for information on instructions in other subsets, see the tables of contents and indexes of the other volumes.

Definitions

Many of the following definitions assume an in-depth knowledge of the legacy x86 architecture. See "Related Documents" on page xxviii for descriptions of the legacy x86 architecture.

Terms and Notation

In addition to the notation described below, "Opcode-Syntax Notation" in Volume 3 describes notation relating specifically to opcodes.

1011b

A binary value—in this example, a 4-bit value.

F0EAh

A hexadecimal value—in this example a 2-byte value.

[1,2)

A range that includes the left-most value (in this case, 1) but excludes the right-most value (in this case, 2).

7–4

A bit range, from bit 7 to 4, inclusive. The high-order bit is shown first.

128-bit media instructions

Instructions that use the 128-bit XMM registers. These are a combination of the SSE, SSE2 and SSE3 instruction sets.

64-bit media instructions

Instructions that use the 64-bit MMX registers. These are primarily a combination of MMX™ and 3DNow!™

instruction sets, with some additional instructions from the SSE and SSE2 instruction sets.

16-bit mode

Legacy mode or compatibility mode in which a 16-bit address size is active. See *legacy mode* and *compatibility mode*.

32-bit mode

Legacy mode or compatibility mode in which a 32-bit address size is active. See *legacy mode* and *compatibility mode*.

64-bit mode

A submode of *long mode*. In 64-bit mode, the default address size is 64 bits and new features, such as register extensions, are supported for system and application software.

#GP(0)

Notation indicating a general-protection exception (#GP) with error code of 0.

absolute

Said of a displacement that references the base of a code segment rather than an instruction pointer. Contrast with *relative*.

ASID

Address space identifier.

biased exponent

The sum of a floating-point value's exponent and a constant bias for a particular floating-point data type. The bias makes the range of the biased exponent always positive, which allows reciprocation without overflow.

byte

Eight bits.

clear

To write a bit value of 0. Compare *set*.

compatibility mode

A submode of *long mode*. In compatibility mode, the default address size is 32 bits, and legacy 16-bit and 32-bit applications run without modification.

commit

To irreversibly write, in program order, an instruction's result to software-visible storage, such as a register (including flags), the data cache, an internal write buffer, or memory.

CPL

Current privilege level.

CR0–CR4

A register range, from register CR0 through CR4, inclusive, with the low-order register first.

CR0.PE = 1

Notation indicating that the PE bit of the CR0 register has a value of 1.

direct

Referencing a memory location whose address is included in the instruction's syntax as an immediate operand. The address may be an absolute or relative address. Compare *indirect*.

dirty data

Data held in the processor's caches or internal buffers that is more recent than the copy held in main memory.

displacement

A signed value that is added to the base of a segment (absolute addressing) or an instruction pointer (relative addressing). Same as *offset*.

doubleword

Two words, or four bytes, or 32 bits.

double quadword

Eight words, or 16 bytes, or 128 bits. Also called *octword*.

DS:rSI

The contents of a memory location whose segment address is in the DS register and whose offset relative to that segment is in the rSI register.

EFER.LME = 0

Notation indicating that the LME bit of the EFER register has a value of 0.

effective address size

The address size for the current instruction after accounting for the default address size and any address-size override prefix.

effective operand size

The operand size for the current instruction after accounting for the default operand size and any operand-size override prefix.

element

See *vector*.

exception

An abnormal condition that occurs as the result of executing an instruction. The processor's response to an exception depends on the type of the exception. For all exceptions except 128-bit media SIMD floating-point exceptions and x87 floating-point exceptions, control is transferred to the handler (or service routine) for that exception, as defined by the exception's vector. For floating-point exceptions defined by the IEEE 754 standard, there are both masked and unmasked responses. When unmasked, the exception handler is called, and when masked, a default response is provided instead of calling the handler.

FF /0

Notation indicating that FF is the first byte of an opcode, and a subopcode in the ModR/M byte has a value of 0.

flush

An often ambiguous term meaning (1) writeback, if modified, and invalidate, as in "flush the cache line," or (2) invalidate, as in "flush the pipeline," or (3) change a value, as in "flush to zero."

GDT

Global descriptor table.

GIF

Global interrupt flag.

IDT
Interrupt descriptor table.

IGN
Ignore. Field is ignored.

indirect
Referencing a memory location whose address is in a register or other memory location. The address may be an absolute or relative address. Compare *direct*.

IRB
The virtual-8086 mode interrupt-redirection bitmap.

IST
The long-mode interrupt-stack table.

IVT
The real-address mode interrupt-vector table.

LDT
Local descriptor table.

legacy x86
The legacy x86 architecture. See "Related Documents" on page xxviii for descriptions of the legacy x86 architecture.

legacy mode
An operating mode of the AMD64 architecture in which existing 16-bit and 32-bit applications and operating systems run without modification. A processor implementation of the AMD64 architecture can run in either *long mode* or *legacy mode*. Legacy mode has three submodes, *real mode*, *protected mode*, and *virtual-8086 mode*.

long mode
An operating mode unique to the AMD64 architecture. A processor implementation of the AMD64 architecture can run in either *long mode* or *legacy mode*. Long mode has two submodes, *64-bit mode* and *compatibility mode*.

lsb
Least-significant bit.

LSB

Least-significant byte.

main memory

Physical memory, such as RAM and ROM (but not cache memory) that is installed in a particular computer system.

mask

(1) A control bit that prevents the occurrence of a floating-point exception from invoking an exception-handling routine. (2) A field of bits used for a control purpose.

MBZ

Must be zero. If software attempts to set an MBZ bit to 1, a general-protection exception (#GP) occurs.

memory

Unless otherwise specified, *main memory.*

ModRM

A byte following an instruction opcode that specifies address calculation based on mode (Mod), register (R), and memory (M) variables.

moffset

A 16, 32, or 64-bit offset that specifies a memory operand directly, without using a ModRM or SIB byte.

msb

Most-significant bit.

MSB

Most-significant byte.

multimedia instructions

A combination of *128-bit media instructions* and *64-bit media instructions*.

octword

Same as *double quadword.*

offset

Same as *displacement*.

overflow

The condition in which a floating-point number is larger in magnitude than the largest, finite, positive or negative number that can be represented in the data-type format being used.

packed

See *vector*.

PAE

Physical-address extensions.

physical memory

Actual memory, consisting of *main memory* and cache.

probe

A check for an address in a processor's caches or internal buffers. *External probes* originate outside the processor, and *internal probes* originate within the processor.

protected mode

A submode of *legacy mode*.

quadword

Four words, or eight bytes, or 64 bits.

RAZ

Read as zero (0), regardless of what is written.

real-address mode

See *real mode*.

real mode

A short name for *real-address mode,* a submode of *legacy mode*.

relative

Referencing with a displacement (also called offset) from an instruction pointer rather than the base of a code segment. Contrast with *absolute*.

reserved

Fields marked as reserved may be used at some future time.

To preserve compatibility with future processors, reserved fields require special handling when read or written by software.

Reserved fields may be further qualified as MBZ, RAZ, SBZ or IGN (see definitions).

Software must not depend on the state of a reserved field, nor upon the ability of such fields to return to a previously written state.

If a reserved field is not marked with one of the above qualifiers, software must not change the state of that field; it must reload that field with the same values returned from a prior read.

REX

An instruction prefix that specifies a 64-bit operand size and provides access to additional registers.

RIP-relative addressing

Addressing relative to the 64-bit RIP instruction pointer.

set

To write a bit value of 1. Compare *clear*.

SIB

A byte following an instruction opcode that specifies address calculation based on scale (S), index (I), and base (B).

SIMD

Single instruction, multiple data. See *vector*.

SSE

Streaming SIMD extensions instruction set. See *128-bit media instructions* and *64-bit media instructions*.

SSE2

Extensions to the SSE instruction set. See *128-bit media instructions* and *64-bit media instructions*.

SSE3

Further extensions to the SSE instruction set. See *128-bit media instructions*.

sticky bit

A bit that is set or cleared by hardware and that remains in that state until explicitly changed by software.

TOP

The x87 top-of-stack pointer.

TSS

Task-state segment.

underflow

The condition in which a floating-point number is smaller in magnitude than the smallest nonzero, positive or negative number that can be represented in the data-type format being used.

vector

(1) A set of integer or floating-point values, called *elements*, that are packed into a single operand. Most of the 128-bit and 64-bit media instructions use vectors as operands. Vectors are also called *packed* or *SIMD* (single-instruction multiple-data) operands.

(2) An index into an interrupt descriptor table (IDT), used to access exception handlers. Compare *exception*.

virtual-8086 mode

A submode of *legacy mode*.

VMCB

Virtual machine control block.

VMM

Virtual machine monitor.

word

Two bytes, or 16 bits.

x86

See *legacy x86*.

Registers

In the following list of registers, the names are used to refer either to a given register or to the contents of that register:

AH–DH

The high 8-bit AH, BH, CH, and DH registers. Compare *AL–DL.*

AL–DL

The low 8-bit AL, BL, CL, and DL registers. Compare *AH–DH.*

AL–r15B

The low 8-bit AL, BL, CL, DL, SIL, DIL, BPL, SPL, and R8B–R15B registers, available in 64-bit mode.

BP

Base pointer register.

CRn

Control register number *n*.

CS

Code segment register.

eAX–eSP

The 16-bit AX, BX, CX, DX, DI, SI, BP, and SP registers or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, and ESP registers. Compare *rAX–rSP.*

EFER

Extended features enable register.

eFLAGS

16-bit or 32-bit flags register. Compare *rFLAGS*.

EFLAGS

32-bit (extended) flags register.

eIP

16-bit or 32-bit instruction-pointer register. Compare *rIP.*

EIP

32-bit (extended) instruction-pointer register.

FLAGS

16-bit flags register.

GDTR

Global descriptor table register.

GPRs

General-purpose registers. For the 16-bit data size, these are AX, BX, CX, DX, DI, SI, BP, and SP. For the 32-bit data size, these are EAX, EBX, ECX, EDX, EDI, ESI, EBP, and ESP. For the 64-bit data size, these include RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, and R8–R15.

IDTR

Interrupt descriptor table register.

IP

16-bit instruction-pointer register.

LDTR

Local descriptor table register.

MSR

Model-specific register.

r8–r15

The 8-bit R8B–R15B registers, or the 16-bit R8W–R15W registers, or the 32-bit R8D–R15D registers, or the 64-bit R8–R15 registers.

rAX–rSP

The 16-bit AX, BX, CX, DX, DI, SI, BP, and SP registers, or the 32-bit EAX, EBX, ECX, EDX, EDI, ESI, EBP, and ESP registers, or the 64-bit RAX, RBX, RCX, RDX, RDI, RSI, RBP, and RSP registers. Replace the placeholder *r* with nothing for 16-bit size, "E" for 32-bit size, or "R" for 64-bit size.

RAX

64-bit version of the EAX register.

RBP

64-bit version of the EBP register.

RBX

64-bit version of the EBX register.

RCX

64-bit version of the ECX register.

RDI
: 64-bit version of the EDI register.

RDX
: 64-bit version of the EDX register.

rFLAGS
: 16-bit, 32-bit, or 64-bit flags register. Compare *RFLAGS*.

RFLAGS
: 64-bit flags register. Compare *rFLAGS*.

rIP
: 16-bit, 32-bit, or 64-bit instruction-pointer register. Compare *RIP*.

RIP
: 64-bit instruction-pointer register.

RSI
: 64-bit version of the ESI register.

RSP
: 64-bit version of the ESP register.

SP
: Stack pointer register.

SS
: Stack segment register.

TPR
: Task priority register (CR8), a new register introduced in the AMD64 architecture to speed interrupt management.

TR
: Task register.

Endian Order

The x86 and AMD64 architectures address memory using little-endian byte-ordering. Multibyte values are stored with their least-significant byte at the lowest byte address, and they are illustrated with their least significant byte at the right side. Strings are illustrated in reverse order, because the addresses of their bytes increase from right to left.

Related Documents

- Peter Abel, *IBM PC Assembly Language and Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- Rakesh Agarwal, *80x86 Architecture & Programming: Volume II*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
- AMD, *AMD-K6™ MMX™ Enhanced Processor Multimedia Technology*, Sunnyvale, CA, 2000.
- AMD, *3DNow!™ Technology Manual*, Sunnyvale, CA, 2000.
- AMD, *AMD Extensions to the 3DNow!™ and MMX™ Instruction Sets*, Sunnyvale, CA, 2000.
- Don Anderson and Tom Shanley, *Pentium Processor System Architecture*, Addison-Wesley, New York, 1995.
- Nabajyoti Barkakati and Randall Hyde, *Microsoft Macro Assembler Bible*, Sams, Carmel, Indiana, 1992.
- Barry B. Brey, *8086/8088, 80286, 80386, and 80486 Assembly Language Programming*, Macmillan Publishing Co., New York, 1994.
- Barry B. Brey, *Programming the 80286, 80386, 80486, and Pentium Based Personal Computer*, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- Ralf Brown and Jim Kyle, *PC Interrupts,* Addison-Wesley, New York, 1994.
- Penn Brumm and Don Brumm, *80386/80486 Assembly Language Programming*, Windcrest McGraw-Hill, 1993.
- Geoff Chappell, *DOS Internals,* Addison-Wesley, New York, 1994.
- Chips and Technologies, Inc. *Super386 DX Programmer's Reference Manual*, Chips and Technologies, Inc., San Jose, 1992.
- John Crawford and Patrick Gelsinger, *Programming the 80386*, Sybex, San Francisco, 1987.
- Cyrix Corporation, *5x86 Processor BIOS Writer's Guide*, Cyrix Corporation, Richardson, TX, 1995.
- Cyrix Corporation, *M1 Processor Data Book*, Cyrix Corporation, Richardson, TX, 1996.
- Cyrix Corporation, *MX Processor MMX Extension Opcode Table*, Cyrix Corporation, Richardson, TX, 1996.
- Cyrix Corporation, *MX Processor Data Book*, Cyrix Corporation, Richardson, TX, 1997.

- Ray Duncan, *Extending DOS: A Programmer's Guide to Protected-Mode DOS*, Addison Wesley, NY, 1991.
- William B. Giles, *Assembly Language Programming for the Intel 80xxx Family*, Macmillan, New York, 1991.
- Frank van Gilluwe, *The Undocumented PC,* Addison-Wesley, New York, 1994.
- John L. Hennessy and David A. Patterson, *Computer Architecture*, Morgan Kaufmann Publishers, San Mateo, CA, 1996.
- Thom Hogan, *The Programmer's PC Sourcebook*, Microsoft Press, Redmond, WA, 1991.
- Hal Katircioglu, *Inside the 486, Pentium, and Pentium Pro*, Peer-to-Peer Communications, Menlo Park, CA, 1997.
- IBM Corporation, *486SLC Microprocessor Data Sheet*, IBM Corporation, Essex Junction, VT, 1993.
- IBM Corporation, *486SLC2 Microprocessor Data Sheet*, IBM Corporation, Essex Junction, VT, 1993.
- IBM Corporation, *80486DX2 Processor Floating Point Instructions*, IBM Corporation, Essex Junction, VT, 1995.
- IBM Corporation, *80486DX2 Processor BIOS Writer's Guide*, IBM Corporation, Essex Junction, VT, 1995.
- IBM Corporation, *Blue Lightening 486DX2 Data Book*, IBM Corporation, Essex Junction, VT, 1994.
- Institute of Electrical and Electronics Engineers, *IEEE Standard for Binary Floating-Point Arithmetic*, ANSI/IEEE Std 754-1985.
- Institute of Electrical and Electronics Engineers, *IEEE Standard for Radix-Independent Floating-Point Arithmetic*, ANSI/IEEE Std 854-1987.
- Muhammad Ali Mazidi and Janice Gillispie Mazidi, *80X86 IBM PC and Compatible Computers*, Prentice-Hall, Englewood Cliffs, NJ, 1997.
- Hans-Peter Messmer, *The Indispensable Pentium Book,* Addison-Wesley, New York, 1995.
- Karen Miller, *An Assembly Language Introduction to Computer Architecture: Using the Intel Pentium*, Oxford University Press, New York, 1999.
- Stephen Morse, Eric Isaacson, and Douglas Albert, *The 80386/387 Architecture*, John Wiley & Sons, New York, 1987.

- NexGen Inc., *Nx586 Processor Data Book*, NexGen Inc., Milpitas, CA, 1993.
- NexGen Inc., *Nx686 Processor Data Book*, NexGen Inc., Milpitas, CA, 1994.
- Bipin Patwardhan, *Introduction to the Streaming SIMD Extensions in the Pentium III*, www.x86.org/articles/sse_pt1/simd1.htm, June, 2000.
- Peter Norton, Peter Aitken, and Richard Wilton, *PC Programmer's Bible,* Microsoft Press, Redmond, WA, 1993.
- *PharLap 386|ASM Reference Manual*, Pharlap, Cambridge MA, 1993.
- *PharLap TNT DOS-Extender Reference Manual*, Pharlap, Cambridge MA, 1995.
- Sen-Cuo Ro and Sheau-Chuen Her, *i386/i486 Advanced Programming*, Van Nostrand Reinhold, New York, 1993.
- Jeffrey P. Royer, *Introduction to Protected Mode Programming*, course materials for an onsite class, 1992.
- Tom Shanley, *Protected Mode System Architecture*, Addison Wesley, NY, 1996.
- SGS-Thomson Corporation, *80486DX Processor SMM Programming Manual*, SGS-Thomson Corporation, 1995.
- Walter A. Triebel, *The 80386DX Microprocessor*, Prentice-Hall, Englewood Cliffs, NJ, 1992.
- John Wharton, *The Complete x86*, MicroDesign Resources, Sebastopol, California, 1994.
- Web sites and newsgroups:
  - www.amd.com
  - news.comp.arch
  - news.comp.lang.asm.x86
  - news.intel.microprocessors
  - news.microsoft

1 128-Bit Media Instruction Reference

This chapter describes the function, mnemonic syntax, opcodes, affected flags of the 128-bit media instructions and the possible exceptions they generate. These instructions load, store, or operate on data located in 128-bit XMM registers. Most of the instructions operate in parallel on sets of packed elements called *vectors*, although a few operate on scalars. These instructions define both integer and floating-point operations. They include the SSE, SSE2 and SSE3 instructions.

Each instruction that performs a vector (packed) operation is illustrated with a diagram. Figure 1-1 on page 1 shows the conventions used in these diagrams. The particular diagram shows the PSLLW (packed shift left logical words) instruction.

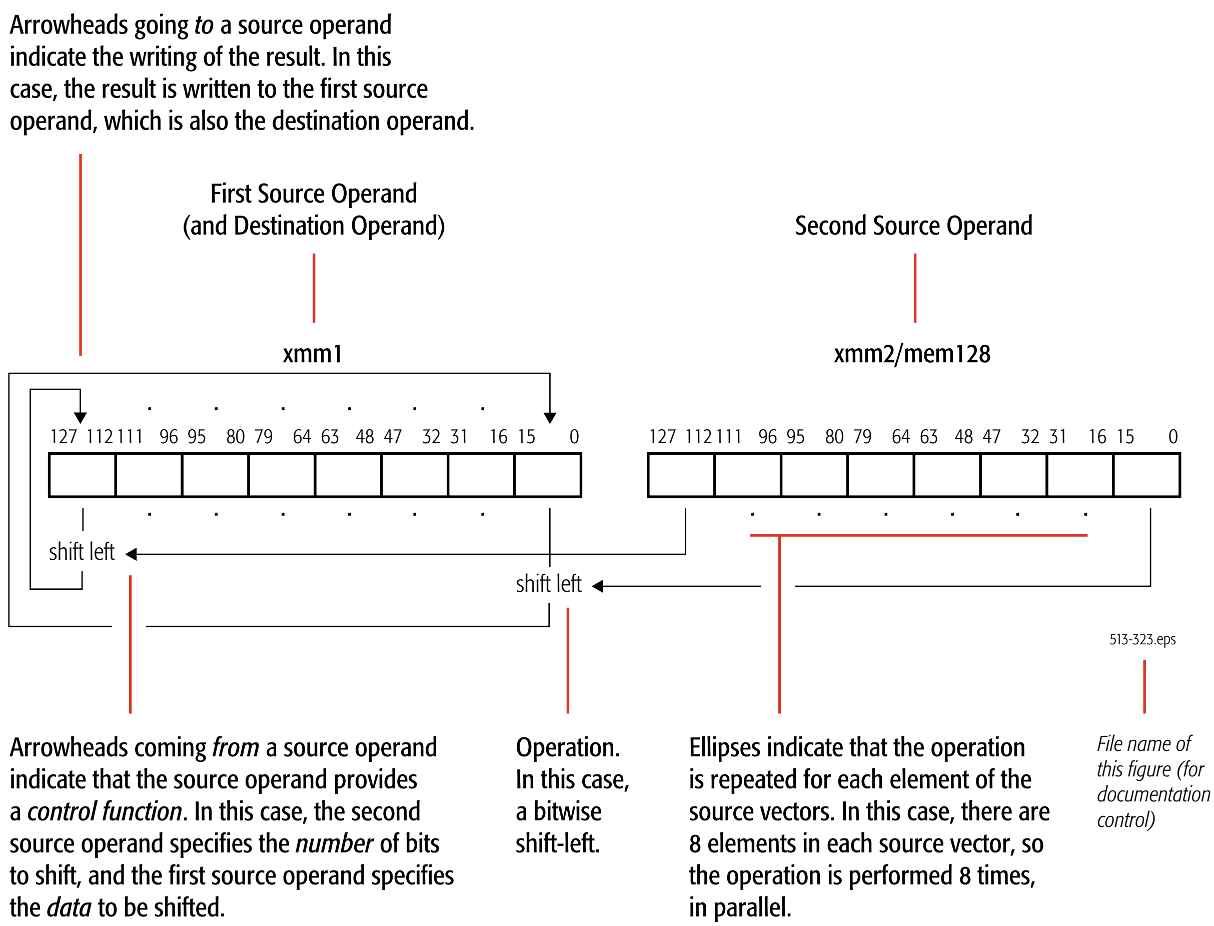


Figure 1-1. Diagram Conventions for 128-Bit Media Instructions

Gray areas in diagrams indicate unmodified operand bits.

The 128-bit media instructions are useful in high-performance applications that operate on blocks of data. Because each instruction can independently and simultaneously perform a single operation on multiple elements of a vector, the instructions are classified as *single-instruction, multiple-data* (SIMD) instructions. A few 128-bit media instructions convert operands in XMM registers to operands in GPR, MMX™, or x87 registers (or vice versa), or save or restore XMM state.

Hardware support for a specific 128-bit media instruction depends on the presence of at least one of the following CPUID functions:

- FXSAVE and FXRSTOR, indicated by EDX bit 24 returned by CPUID standard function 1 and extended function 8000_0001h.
- SSE, indicated by EDX bit 25 returned by CPUID standard function 1.
- SSE2, indicated by EDX bit 26 returned by CPUID standard function 1.
- SSE3, indicated by ECX bit 0 returned by CPUID standard function 1.

The 128-bit media instructions can be used in legacy mode or long mode. Their use in long mode is available if the following CPUID function is set:

- Long Mode, indicated by EDX bit 29 returned by CPUID extended function 8000_0001h.

Compilation of 128-bit media programs for execution in 64-bit mode offers four primary advantages: access to the eight extended XMM registers (for a register set consisting of XMM0–XMM15), access to the eight extended, 64-bit general-purpose registers (for a register set consisting of GPR0–GPR15), access to the 64-bit virtual address space, and access to the RIP-relative addressing mode.

For further information, see:

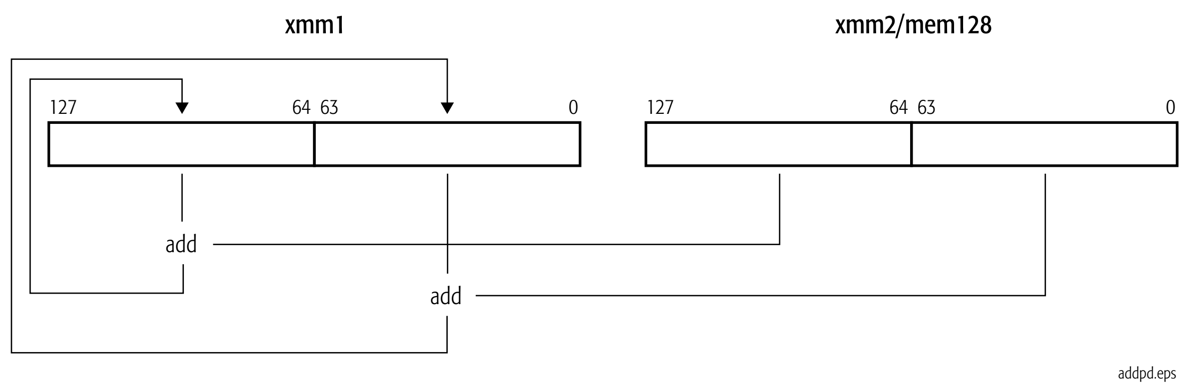
- "128-Bit Media and Scientific Programming" in Volume 1.
- "Summary of Registers and Data Types" in Volume 3.
- "Notation" in Volume 3.
- "Instruction Prefixes" in Volume 3.

ADDPD Add Packed Double-Precision Floating-Point

Adds each packed double-precision floating-point value in the first source operand to the corresponding packed double-precision floating-point value in the second source operand and writes the result of each addition in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ADDPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ADDPD *xmm1*, *xmm2/mem128*	66 0F 58 /*r*	Adds two packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ADDPS, ADDSD, ADDSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

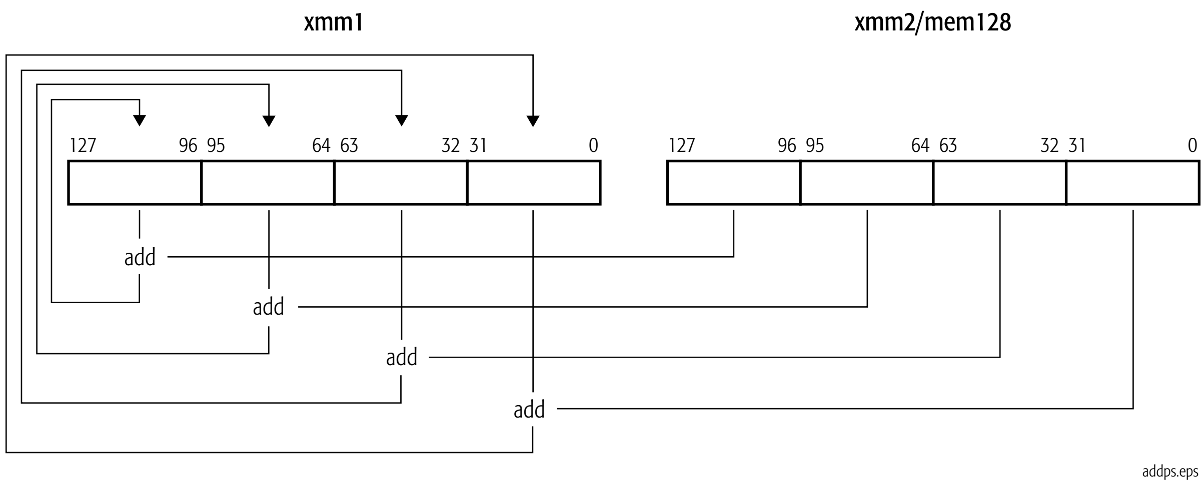
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ADDPS Add Packed Single-Precision Floating-Point

Adds each packed single-precision floating-point value in the first source operand to the corresponding packed single-precision floating-point value in the second source operand and writes the result of each addition in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ADDPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ADDPS *xmm1*, *xmm2/mem128*	0F 58 /*r*	Adds four packed single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ADDPD, ADDSD, ADDSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

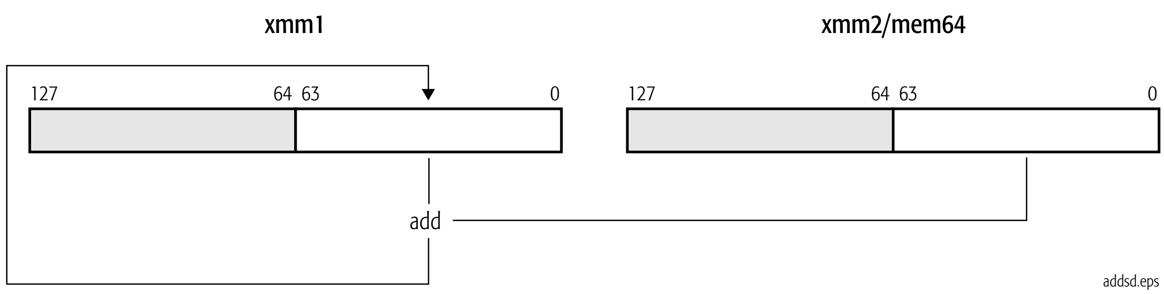
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ADDSD — Add Scalar Double-Precision Floating-Point

Adds the double-precision floating-point value in the low-order quadword of the first source operand to the double-precision floating-point value in the low-order quadword of the second source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location.

The ADDSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
ADDSD *xmm1*, *xmm2/mem64*	F2 0F 58 /*r*	Adds low-order double-precision floating-point values in an XMM register and another XMM register or 64-bit memory location and writes the result in the destination XMM register.



Related Instructions

ADDPD, ADDPS, ADDSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

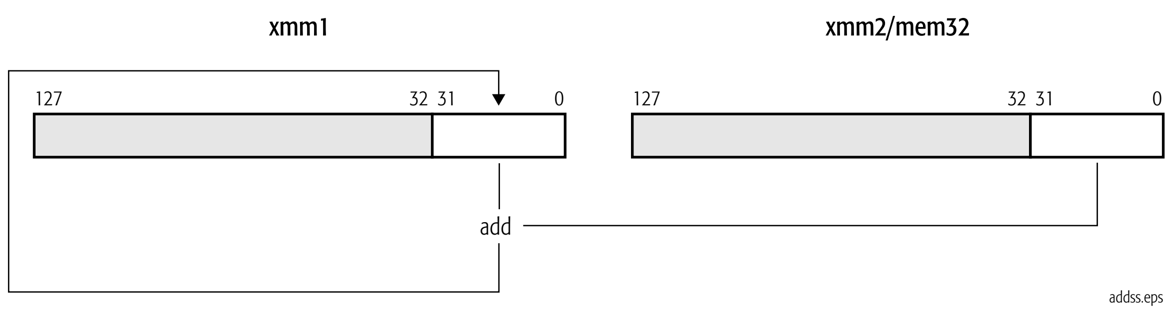
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ADDSS Add Scalar Single-Precision Floating-Point

Adds the single-precision floating-point value in the low-order doubleword of the first source operand to the single-precision floating-point value in the low-order doubleword of the second source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location.

The ADDSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ADDSS *xmm1*, *xmm2/mem32*	F3 0F 58 */r*	Adds low-order single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

ADDPD, ADDPS, ADDSD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

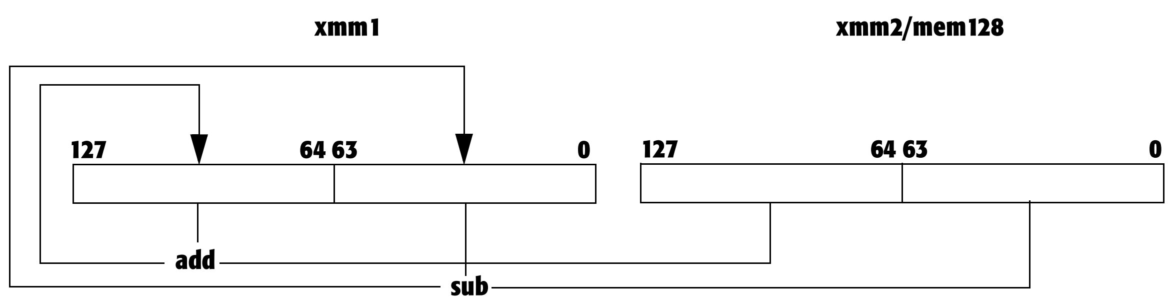
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ADDSUBPD Add and Subtract Packed Double-Precision

Adds the packed double-precision floating-point value in the high 64 bits of the source operand to the double-precision floating-point value in the high 64 bits of the destination operand and stores the sum in the high 64 bits of the destination operand; subtracts the packed double-precision floating-point value in the low 64 bits of the source operand from the low 64 bits of the destination operand and stores the difference in the low 64 bits of the destination operand.

The ADDSUBPD instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
ADDSUBPD *xmm1*, *xmm2/mem128*	66 0F D0 /r	Adds the value in the upper 64 bits of the source operand to the value in the upper 64 bits of the destination operand and stores the result in the upper 64 bits of the destination operand; subtracts the value in the lower 64 bits of the source operand from the value in the lower 64 bits of the destination operand and stores the result in the lower 64 bits of the destination operand.



Related Instructions

ADDSUBPS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.

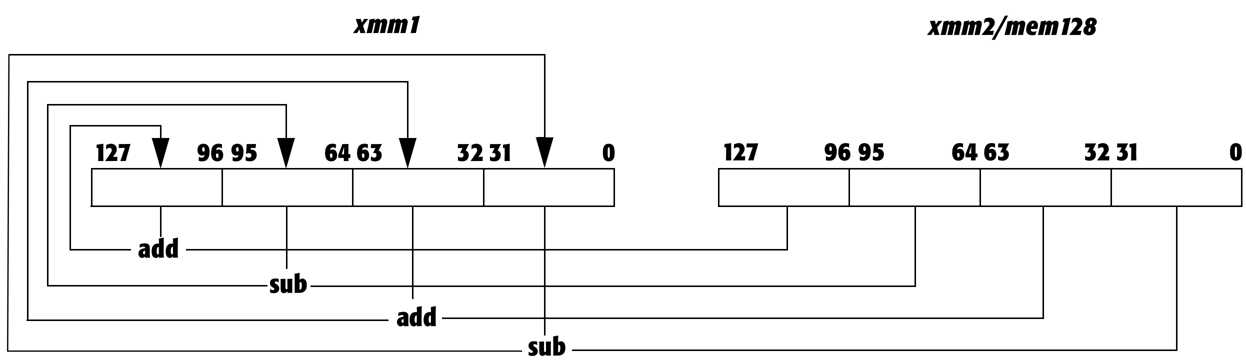
Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ADDSUBPS Add and Subtract Packed Single-Precision

Subtracts the first and third single-precision floating-point values in the source operand from the first and third single-precision floating-point values of the destination operand and stores the result in the first and third values of the destination operand. Simultaneously, the instruction adds the second and fourth single-precision floating-point values in the source operand to the second and fourth single-precision floating-point values in the destination operand and stores the result in the second and fourth values of the destination operand.

The ADDSUBPS instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
ADDSUBPS *xmm1*, *xmm2/mem128*	F2 0F D0 /r	Subtracts the first and third packed single-precision values in the source XMM register or 128-bit memory operand from the corresponding values in the destination XMM register and stores the resulting values in the corresponding positions in the destination register; simultaneously, adds the second and fourth packed single-precision values in the source XMM register or 128-bit memory operand to the corresponding values in the destination register and stores the result in the corresponding positions in the destination register.



Related Instructions

ADDSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS			X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	Anull data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.

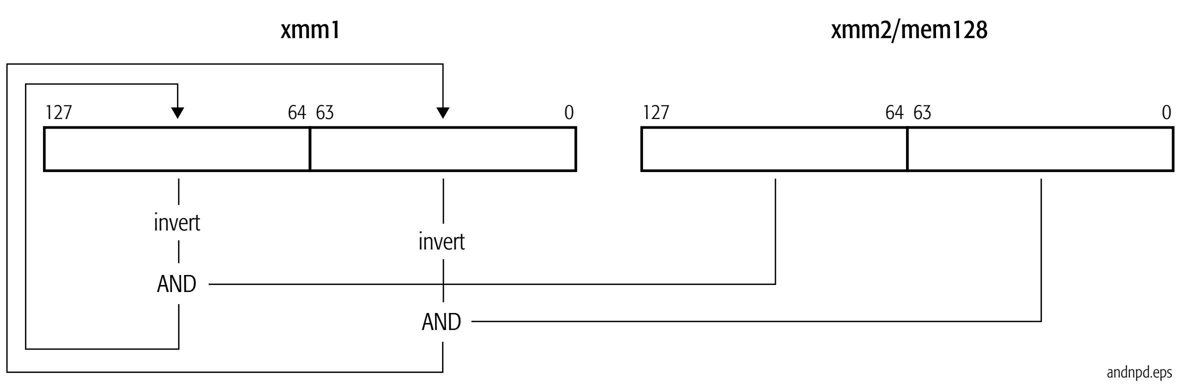
Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ANDNPD Logical Bitwise AND NOT Packed Double-Precision Floating-Point

Performs a bitwise logical AND of the two packed double-precision floating-point values in the second source operand and the one's-complement of the corresponding two packed double-precision floating-point values in the first source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ADDNPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
ANDNPD *xmm1*, *xmm2/mem128*	66 0F 55 */r*	Performs bitwise logical AND NOT of two packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPS, ANDPD, ANDPS, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated byEDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

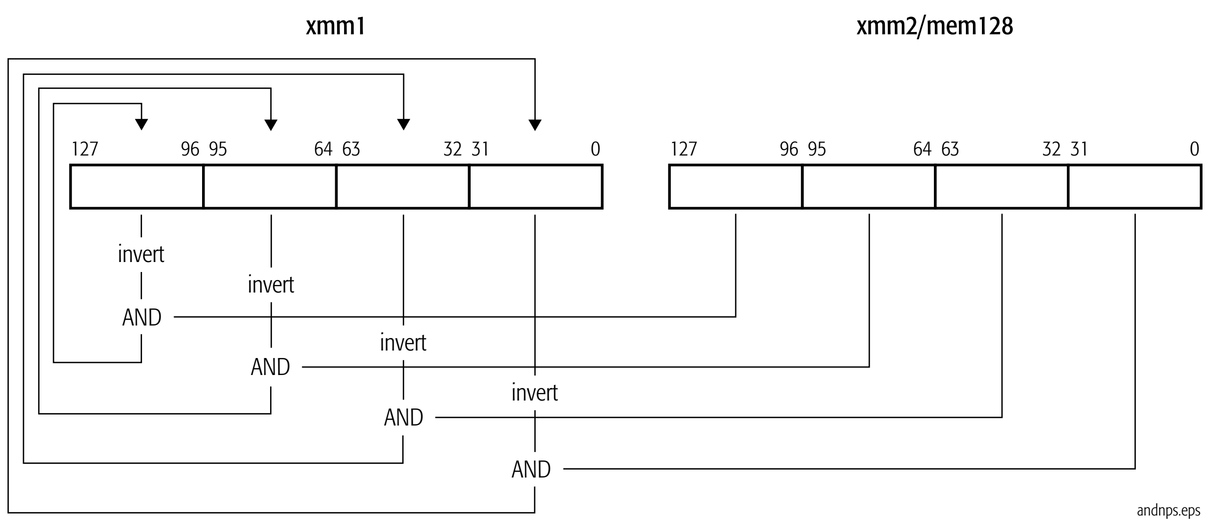
ANDNPS

Logical Bitwise AND NOT Packed Single-Precision Floating-Point

Performs a bitwise logical AND of the four packed single-precision floating-point values in the second source operand and the one's-complement of the corresponding four packed single-precision floating-point values in the first source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ADDNPS instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
ANDNPS *xmm1*, *xmm2/mem128*	0F 55 */r*	Performs bitwise logical AND NOT of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDPD, ANDPS, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

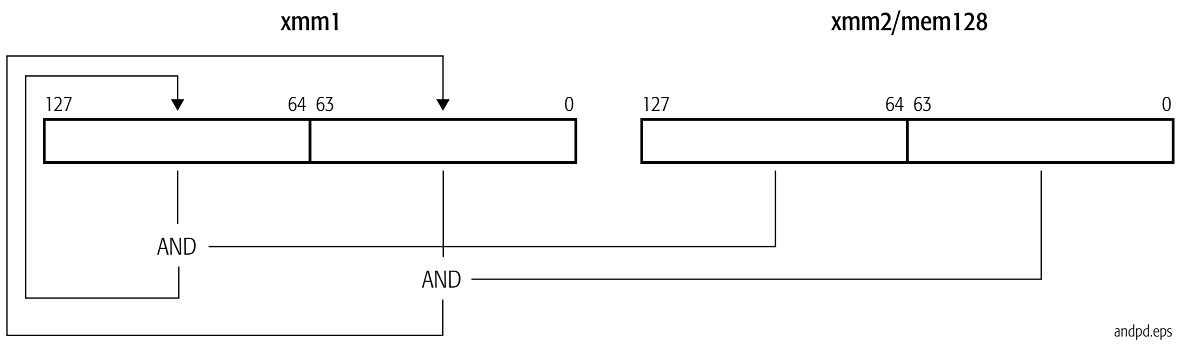
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated byEDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

ANDPD — Logical Bitwise AND Packed Double-Precision Floating-Point

Performs a bitwise logical AND of the two packed double-precision floating-point values in the first source operand and the corresponding two packed double-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ANDPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ANDPD *xmm1*, *xmm2/mem128*	66 0F 54 /*r*	Performs bitwise logical AND of two packed double-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDNPS, ANDPS, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

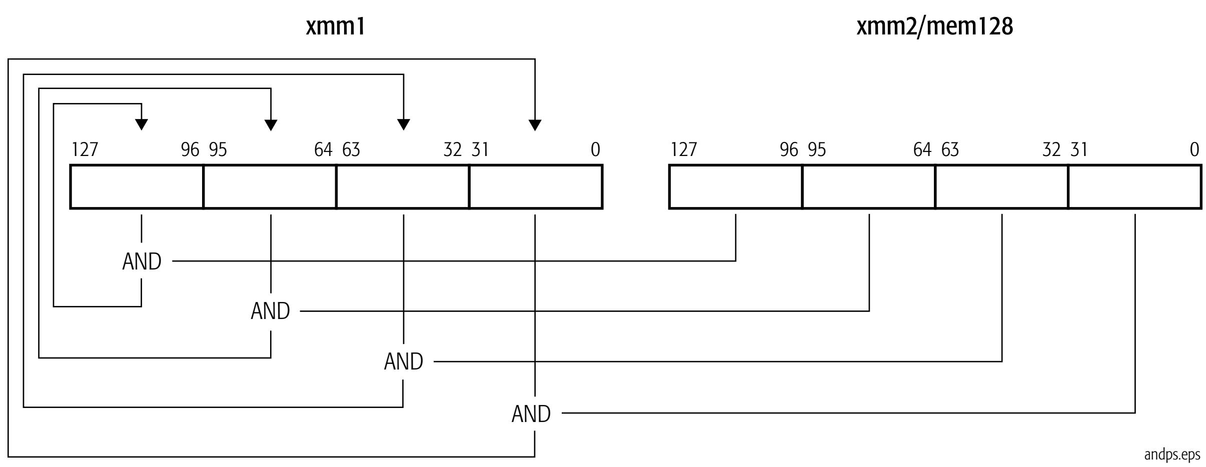
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

ANDPS Logical Bitwise AND Packed Single-Precision Floating-Point

Performs a bitwise logical AND of the four packed single-precision floating-point values in the first source operand and the corresponding four packed single-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ADDPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ANDPS *xmm1*, *xmm2/mem128*	0F 54 /*r*	Performs bitwise logical AND of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDNPS, ANDPD, ORPD, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated byEDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

CMPPD Compare Packed Double-Precision Floating-Point

Compares each of the two packed double-precision floating-point values in the first source operand with the corresponding packed double-precision floating-point value in the second source operand and writes the result of each comparison in the corresponding 64 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1. The result of each compare is a 64-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

Signed compares return TRUE only if both operands are valid numbers, and the numbers have the relation specified by the type of compare. "Ordered" compare returns TRUE if both operands are valid numbers, or FALSE if either operand is a NaN. "Unordered" compare returns TRUE only if one or both operands are NaN, and FALSE otherwise.

QNaN operands generate an Invalid Operation Exception only if the compare type isn't "Equal", "Unequal", "Orderered", or "Unordered". SNaN operands always generate an Invalid Operation Exception (IE).

Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown, together with the directly supported compare operations, in Table 1-1. When swapping operands, the first source XMM register is overwritten by the result.

The CMPPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CMPPD *xmm1*, *xmm2/mem128*, *imm8*	66 0F C2 */r ib*	Compares two pairs of packed double-precision floating-point values in an XMM register and an XMM register or 128-bit memory location.

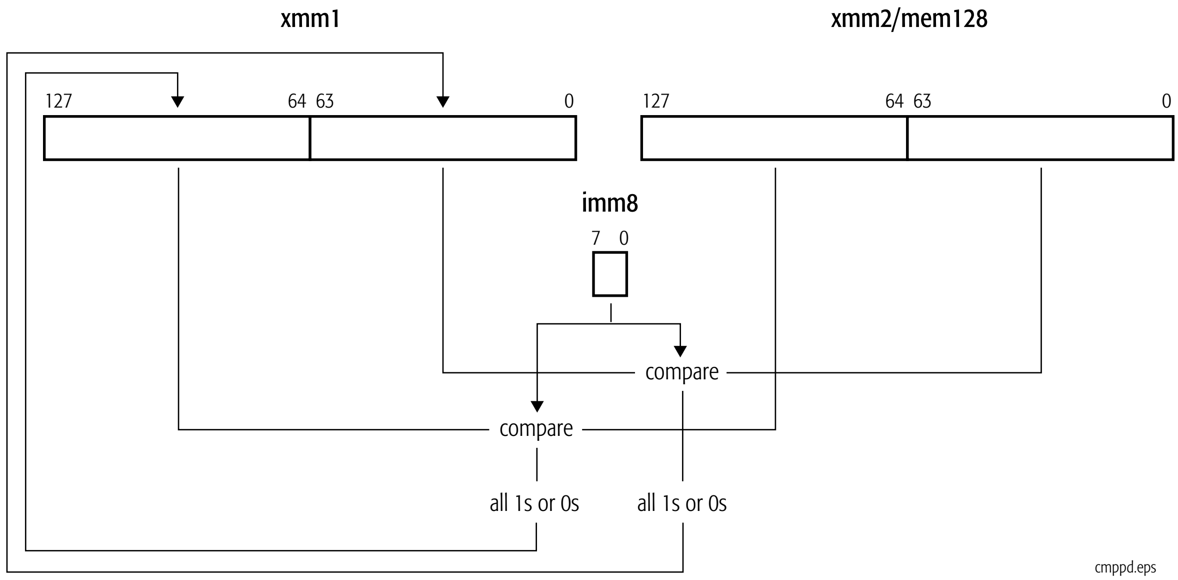


Table 1-1. Immediate Operand Values for Compare Operations

Immediate-Byte Value (bits 2–0)	Compare Operation	Result If NaN Operand	QNaN Operand Causes Invalid Operation Exception
000	Equal	FALSE	No
001	Less than	FALSE	Yes
	Greater than (uses swapped operands)	FALSE	Yes
010	Less than or equal	FALSE	Yes
	Greater than or equal (uses swapped operands)	FALSE	Yes
011	Unordered	TRUE	No
100	Not equal	TRUE	No
101	Not less than	TRUE	Yes
	Not greater than (uses swapped operands)	TRUE	Yes
110	Not less than or equal	TRUE	Yes
	Not greater than or equal (uses swapped operands)	TRUE	Yes
111	Ordered	FALSE	No

Related Instructions

CMPPS, CMPSD, CMPSS, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was a QNaN value, and the comparison does not allow QNaN values (refer to Table 1-1 on page 30).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

CMPPS — Compare Packed Single-Precision Floating-Point

Compares each of the four packed single-precision floating-point values in the first source operand with the corresponding packed single-precision floating-point value in the second source operand and writes the result of each comparison in the corresponding 32 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1 on page 30. The result of each compare is a 32-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

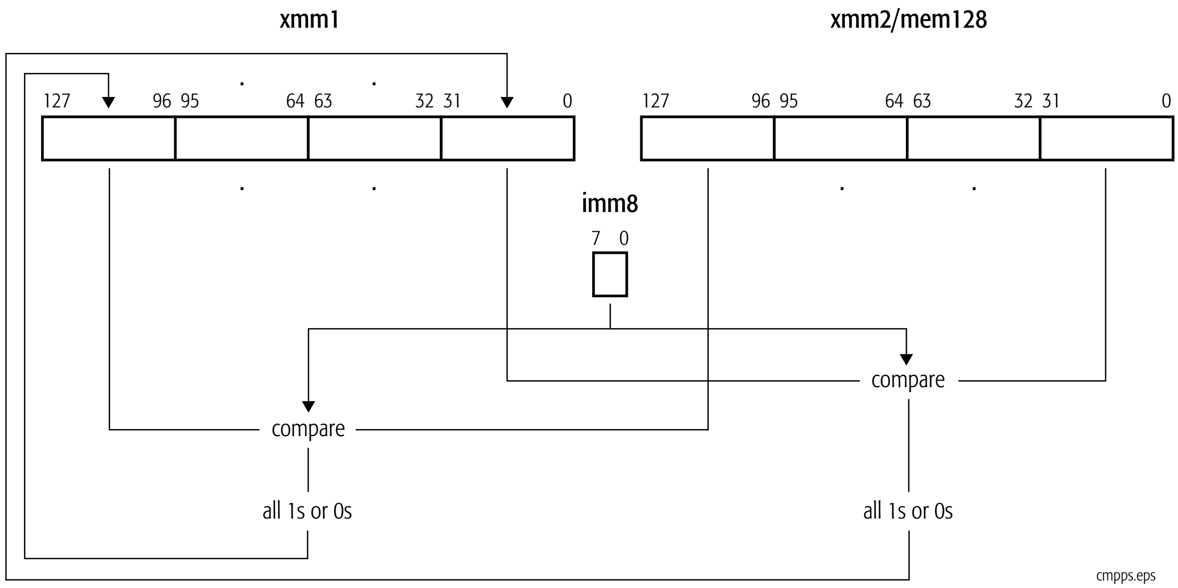
Signed compares return TRUE only if both operands are valid numbers, and the numbers have the relation specified by the type of compare. "Ordered" compare returns TRUE if both operands are valid numbers, or FALSE if either operand is a NaN. "Unordered" compare returns TRUE only if one or both operands are NaN, and FALSE otherwise.

QNaN operands generate an Invalid Operation Exception only if the compare type isn't "Equal", "Unequal", "Orderered", or "Unordered". SNaN operands always generate an Invalid Operation Exception (IE).

Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown in Table 1-1 on page 30. When swapping operands, the first source XMM register is overwritten by the result.

The CMPPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CMPPS *xmm1*, *xmm2/mem128*, *imm8*	0F C2 */r ib*	Compares four pairs of packed single-precision floating-point values in an XMM register and an XMM register or 64-bit memory location.



Related Instructions

CMPPD, CMPSD, CMPSS, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was a QNaN value, and the comparison does not allow QNaN values (refer to Table 1-1 on page 30).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

CMPSD Compare Scalar Double-Precision Floating-Point

Compares the double-precision floating-point value in the low-order 64 bits of the first source operand with the double-precision floating-point value in the low-order 64 bits of the second source operand and writes the result in the low-order 64 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1 on page 30. The result of the compare is a 64-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location. The high-order 64 bits of the destination XMM register are not modified.

Signed compares return TRUE only if both operands are valid numbers, and the numbers have the relation specified by the type of compare. "Ordered" compare returns TRUE if both operands are valid numbers, or FALSE if either operand is a NaN. "Unordered" compare returns TRUE only if one or both operands are NaN, and FALSE otherwise.

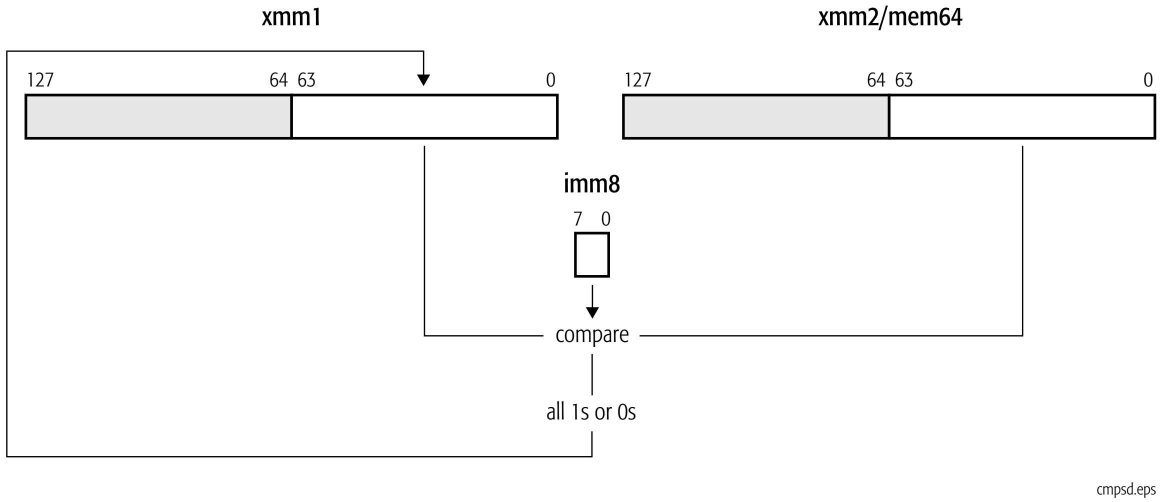
QNaN operands generate an Invalid Operation Exception only if the compare type isn't "Equal", "Unequal", "Orderered", or "Unordered". SNaN operands always generate an Invalid Operation Exception (IE).

Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown in Table 1-1 on page 30. When swapping operands, the first source XMM register is overwritten by the result.

This CMPSD instruction should not be confused with the same-mnemonic CMPSD (compare strings by doubleword) instruction in the general-purpose instruction set. Assemblers can distinguish the instructions by the number and type of operands.

The CMPSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CMPSD *xmm1*, *xmm2/mem64*, *imm8*	F2 0F C2 */r ib*	Compares double-precision floating-point values in an XMM register and an XMM register or 64-bit memory location.



Related Instructions

CMPPD, CMPPS, CMPSS, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was a QNaN value, and the comparison does not allow QNaN values (refer to Table 1-1 on page 30).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

CMPSS Compare Scalar Single-Precision Floating-Point

Compares the single-precision floating-point value in the low-order 32 bits of the first source operand with the single-precision floating-point value in the low-order 32 bits of the second source operand and writes the result in the low-order 32 bits of the destination (first source). The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in Table 1-1 on page 30. The result of the compare is a 32-bit value of all 1s (TRUE) or all 0s (FALSE). The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location. The three high-order doublewords of the destination XMM register are not modified.

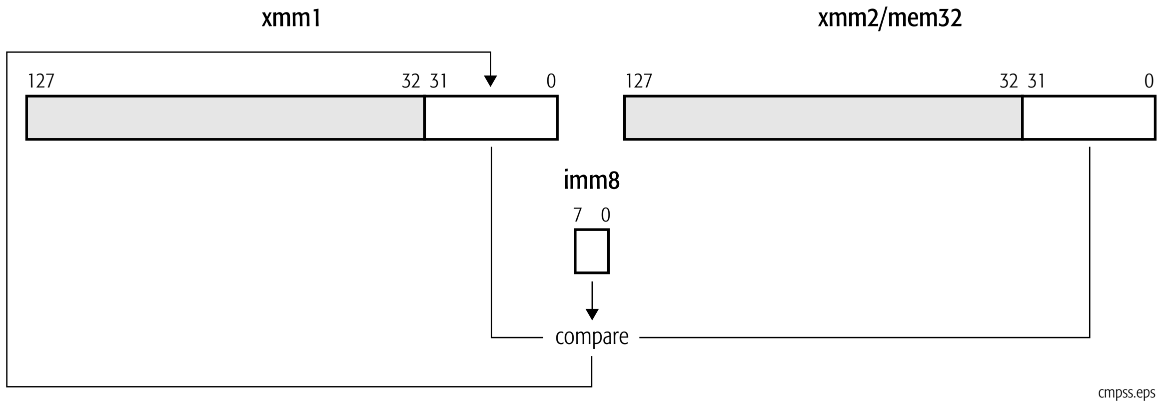
Signed compares return TRUE only if both operands are valid numbers, and the numbers have the relation specified by the type of compare. "Ordered" compare returns TRUE if both operands are valid numbers, or FALSE if either operand is a NaN. "Unordered" compare returns TRUE only if one or both operands are NaN, and FALSE otherwise.

QNaN operands generate an Invalid Operation Exception only if the compare type isn't "Equal", "Unequal", "Orderered", or "Unordered". SNaN operands always generate an Invalid Operation Exception (IE).

Some compare operations that are not directly supported by the immediate-byte encodings can be implemented by swapping the contents of the source and destination operands and then executing the appropriate compare instruction using the swapped values. These additional compare operations are shown in Table 1-1 on page 30. When swapping operands, the first source XMM register is overwritten by the result.

The CMPSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CMPSS *xmm1*, *xmm2/mem32*, *imm8*	F3 0F C2 */r ib*	Compares single-precision floating-point values in an XMM register and an XMM register or 32-bit memory location.



Related Instructions

CMPPD, CMPPS, CMPSD, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was a QNaN value, and the comparison does not allow QNaN values (refer to Table 1-1 on page 30).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

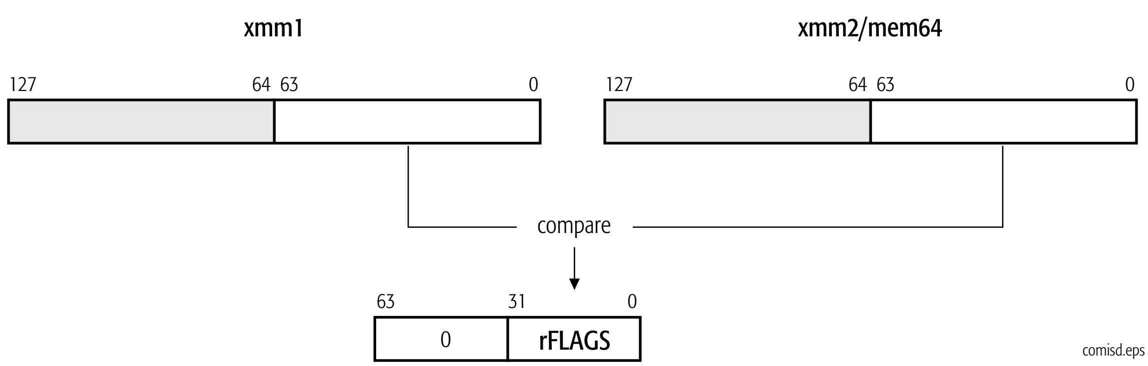
COMISD Compare Ordered Scalar Double-Precision Floating-Point

Compares the double-precision floating-point value in the low-order 64 bits of an XMM register with the double-precision floating-point value in the low-order 64 bits of another XMM register or a 64-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result of the comparison. The OF, AF, and SF bits in rFLAGS are set to zero. The result is unordered if one or both of the operand values is a NaN.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

The COMISD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
COMISD *xmm1*, *xmm2/mem64*	66 0F 2F /*r*	Compares double-precision floating-point values in an XMM register and an XMM register or 64-bit memory location and sets rFLAGS.



Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:

Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set either to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

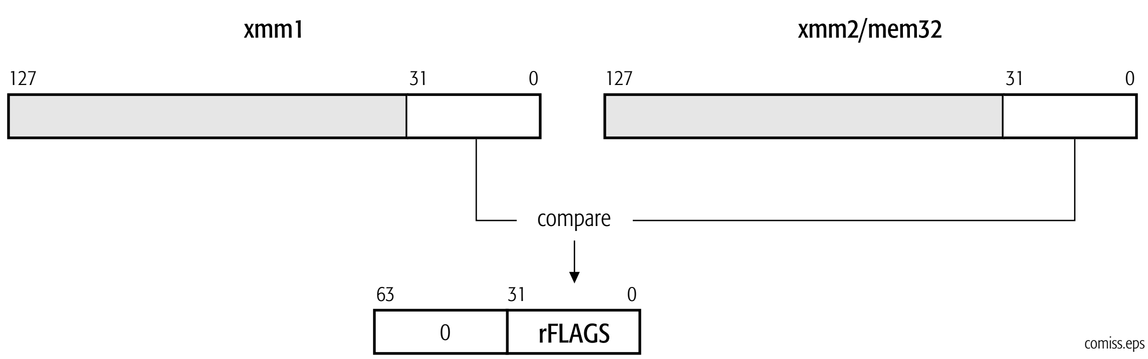
COMISS Compare Ordered Scalar Single-Precision Floating-Point

Performs an ordered comparison of the single-precision floating-point value in the low-order 32 bits of an XMM register with the single-precision floating-point value in the low-order 32 bits of another XMM register or a 32-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result of the comparison. The OF, AF, and SF bits in rFLAGS are set to zero. The result is unordered if one or both of the operand values is a NaN.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

The COMISS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
COMISS *xmm1*, *xmm2/mem32*	0F 2F /*r*	Compares single-precision floating-point values in an XMM register and an XMM register or 32-bit memory location. Sets rFLAGS.



Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISD, UCOMISD, UCOMISS

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:

Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

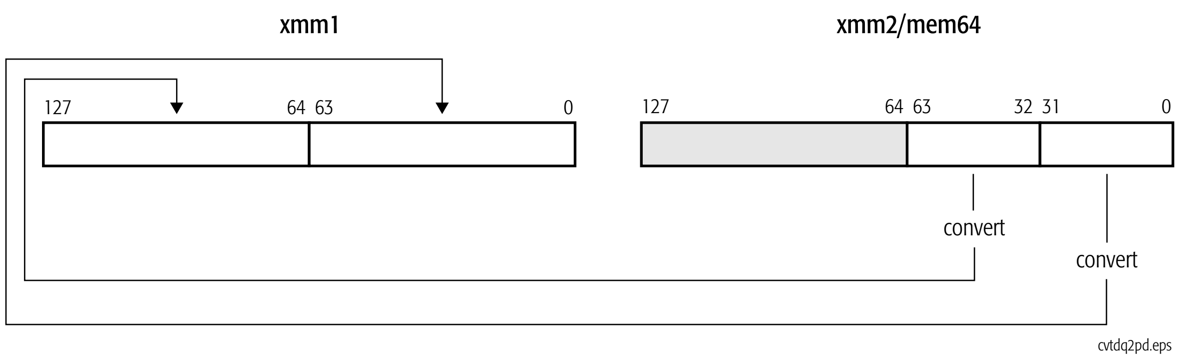
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

CVTDQ2PD Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point

Converts two packed 32-bit signed integer values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed double-precision floating-point values and writes the converted values in another XMM register.

The CVTDQ2PD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CVTDQ2PD *xmm1*, *xmm2/mem64*	F3 0F E6 /r	Converts packed doubleword signed integers in an XMM register or 64-bit memory location to double-precision floating-point values in the destination XMM register.



Related Instructions

CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

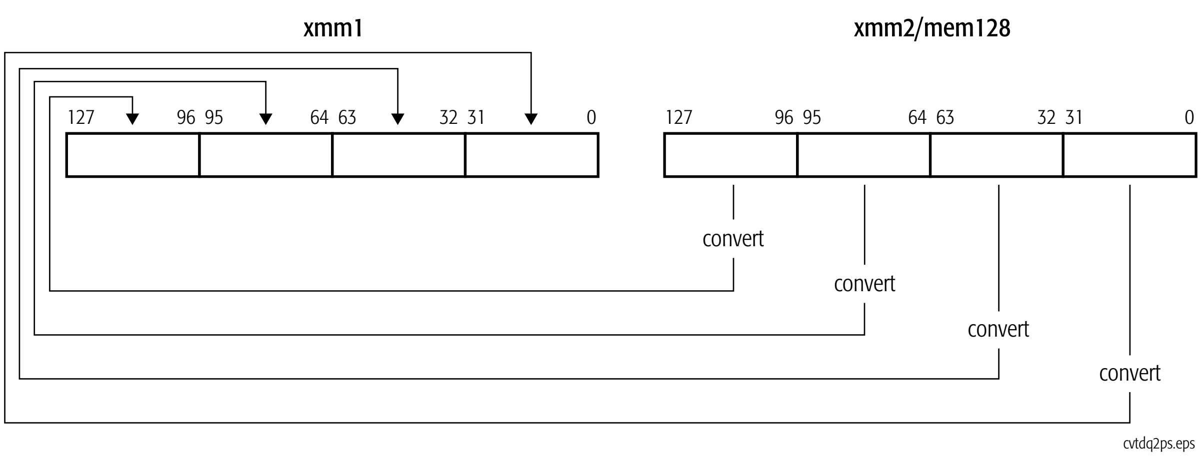
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

CVTDQ2PS Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point

Converts four packed 32-bit signed integer values in an XMM register or a 128-bit memory location to four packed single-precision floating-point values and writes the converted values in another XMM register. If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

The CVTDQ2PS instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTDQ2PS *xmm1*, *xmm2/mem128*	0F 5B /*r*	Converts packed doubleword integer values in an XMM register or 128-bit memory location to packed single-precision floating-point values in the destination XMM register.



Related Instructions

CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	A result coulld not be represented exactly in the destination format.

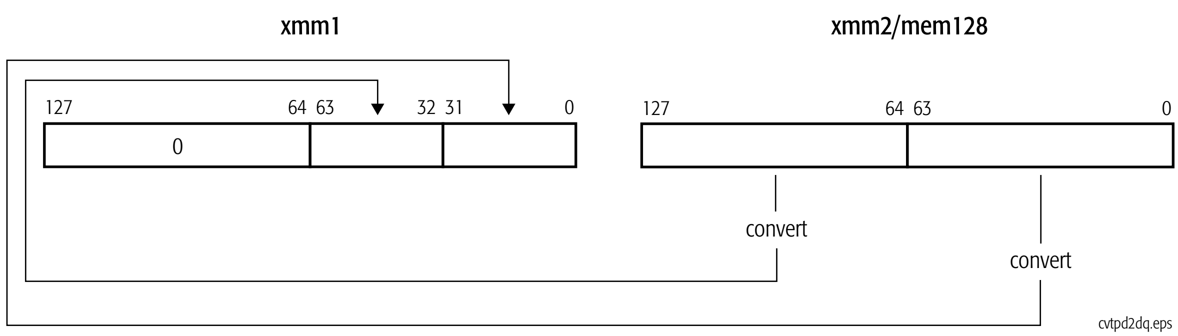
CVTPD2DQ Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers

Converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integers and writes the converted values in the low-order 64 bits of another XMM register. The high-order 64 bits in the destination XMM register are cleared to all 0s.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31}-1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTPD2DQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTPD2DQ *xmm1*, *xmm2/mem128*	F2 0F E6 /r	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integers in the destination XMM register.



Related Instructions

CVTDQ2PD, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

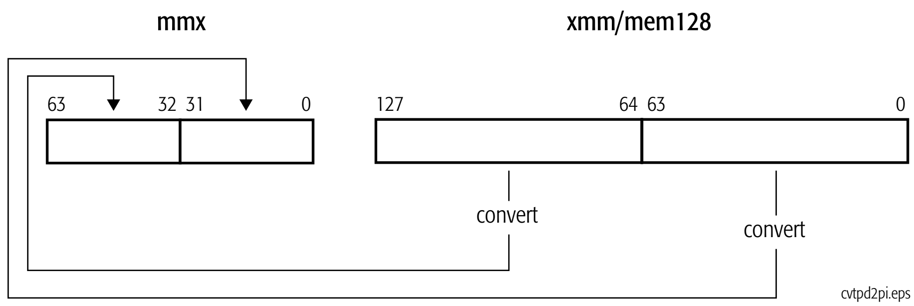
CVTPD2PI Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers

Converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integer values and writes the converted values in an MMX register.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31}-1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTPD2PI instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTPD2PI *mmx, xmm2/mem128*	66 0F 2D */r*	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integers values in the destination MMX register.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Note: *A flag that may be set to one or or cleared to zero is M (modified). Unaffected flags are blank.*															

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

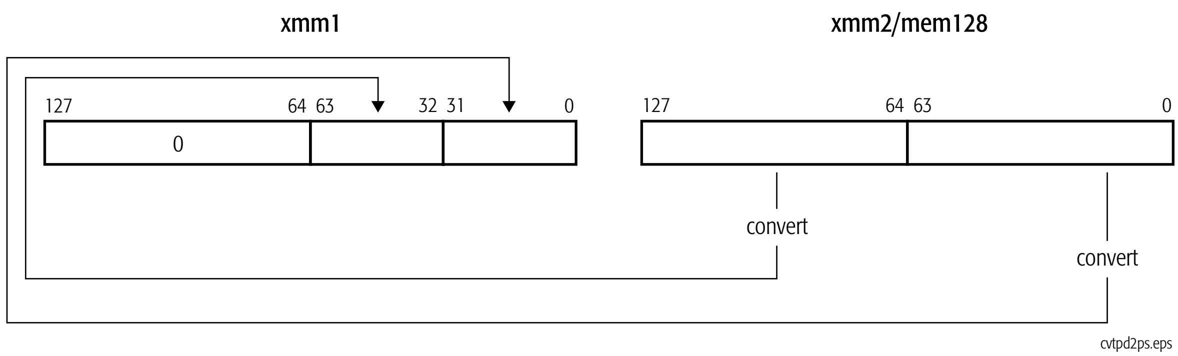
CVTPD2PS Convert Packed Double-Precision Floating-Point to Packed Single-Precision Floating-Point

Converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed single-precision floating-point values and writes the converted values in the low-order 64 bits of another XMM register. The high-order 64 bits in the destination XMM register are cleared to all 0s.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

The CVTPD2PS instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTPD2PS *xmm1*, *xmm2/mem128*	66 0F 5A /*r*	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed single-precision floating-point values in the destination XMM register.



Related Instructions

CVTPS2PD, CVTSD2SS, CVTSS2SD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.

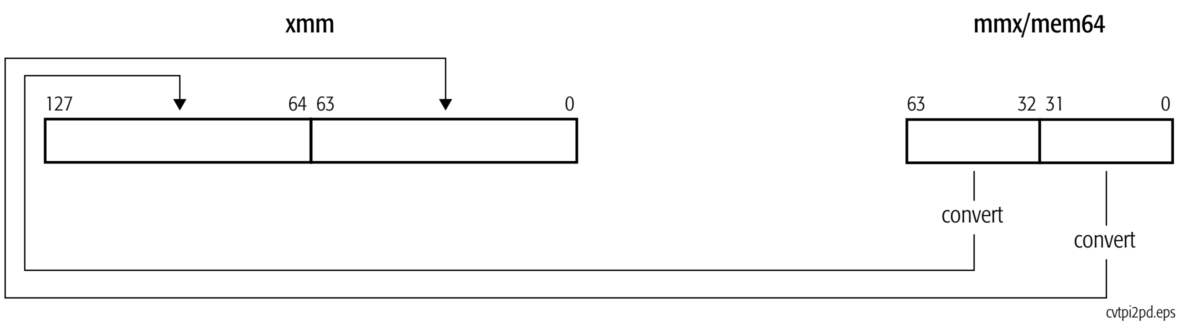
Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

CVTPI2PD Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point

Converts two packed 32-bit signed integer values in an MMX register or a 64-bit memory location to two double-precision floating-point values and writes the converted values in an XMM register.

The CVTPI2PD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CVTPI2PD *xmm*, *mmx/mem64*	66 0F 2A */r*	Converts two packed doubleword integer values in an MMX register or 64-bit memory location to two packed double-precision floating-point values in the destination XMM register.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

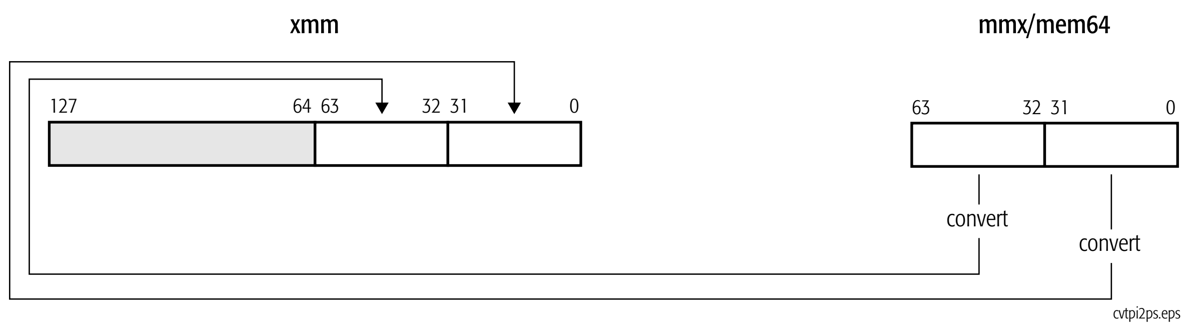
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception was pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

CVTPI2PS Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point

Converts two packed 32-bit signed integer values in an MMX register or a 64-bit memory location to two single-precision floating-point values and writes the converted values in the low-order 64 bits of an XMM register. The high-order 64 bits of the XMM register are not modified.

The CVTPI2PS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CVTPI2PS *xmm*, *mmx/mem64*	0F 2A /*r*	Converts packed doubleword integer values in an MMX register or 64-bit memory location to single-precision floating-point values in the destination XMM register.



Related Instructions

CVTDQ2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception was pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

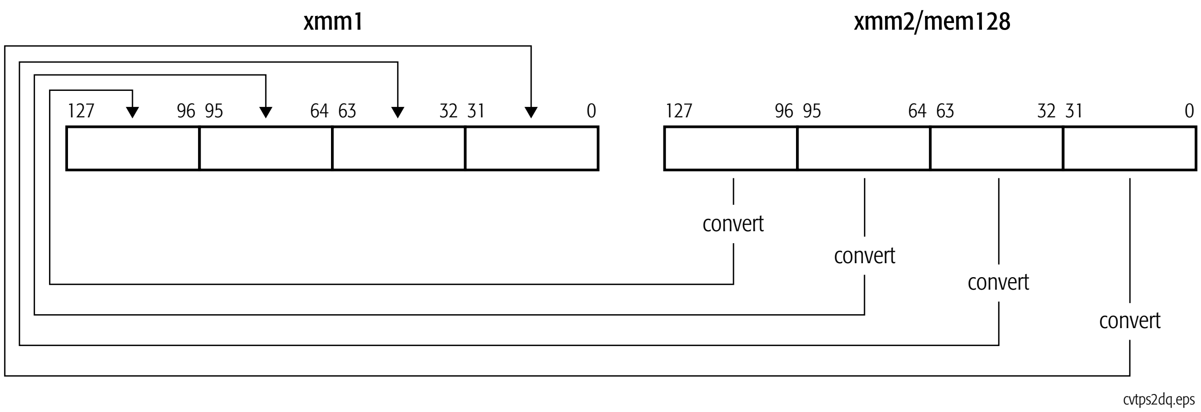
CVTPS2DQ Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers

Converts four packed single-precision floating-point values in an XMM register or a 128-bit memory location to four packed 32-bit signed integer values and writes the converted values in another XMM register.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTPS2DQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTPS2DQ *xmm1*, *xmm2/mem128*	66 0F 5B /*r*	Converts four packed single-precision floating-point values in an XMM register or 128-bit memory location to four packed doubleword integers in the destination XMM register.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				

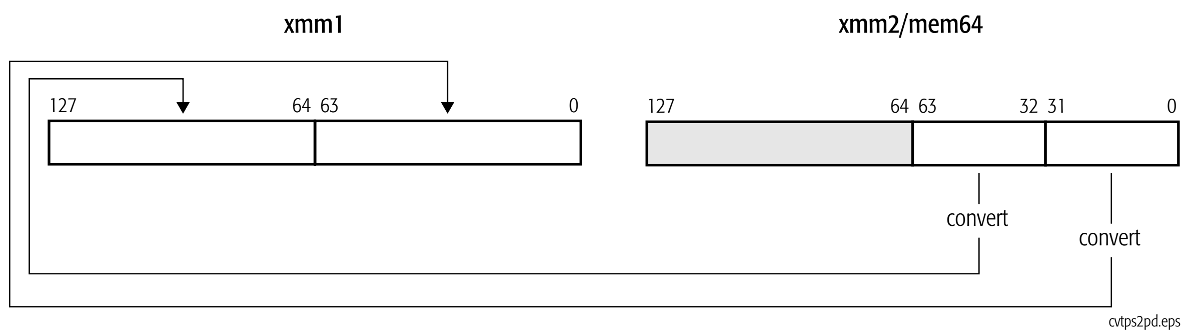
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

CVTPS2PD Convert Packed Single-Precision Floating-Point to Packed Double-Precision Floating-Point

Converts two packed single-precision floating-point values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed double-precision floating-point values and writes the converted values in another XMM register.

The CVTPS2PD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTPS2PD *xmm1*, *xmm2/mem64*	0F 5A /*r*	Converts packed single-precision floating-point values in an XMM register or 64-bit memory location to packed double-precision floating-point values in the destination XMM register.



Related Instructions

CVTPD2PS, CVTSD2SS, CVTSS2SD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

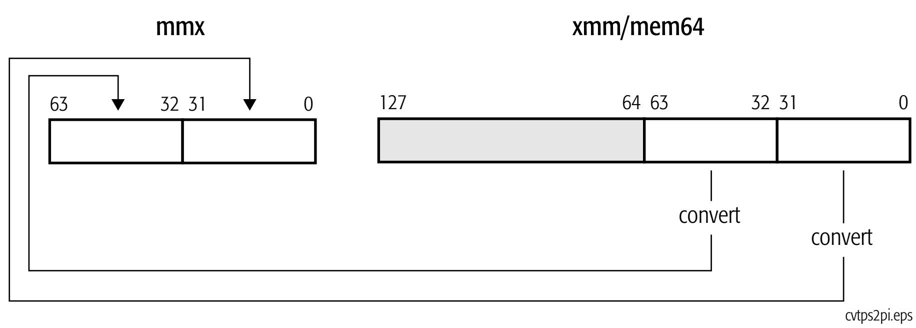
CVTPS2PI Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers

Converts two packed single-precision floating-point values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed 32-bit signed integers and writes the converted values in an MMX register.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTPS2PI instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CVTPS2PI *mmx, xmm/mem64*	0F 2D /*r*	Converts packed single-precision floating-point values in an XMM register or 64-bit memory location to packed doubleword integers in the destination MMX register.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception was pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

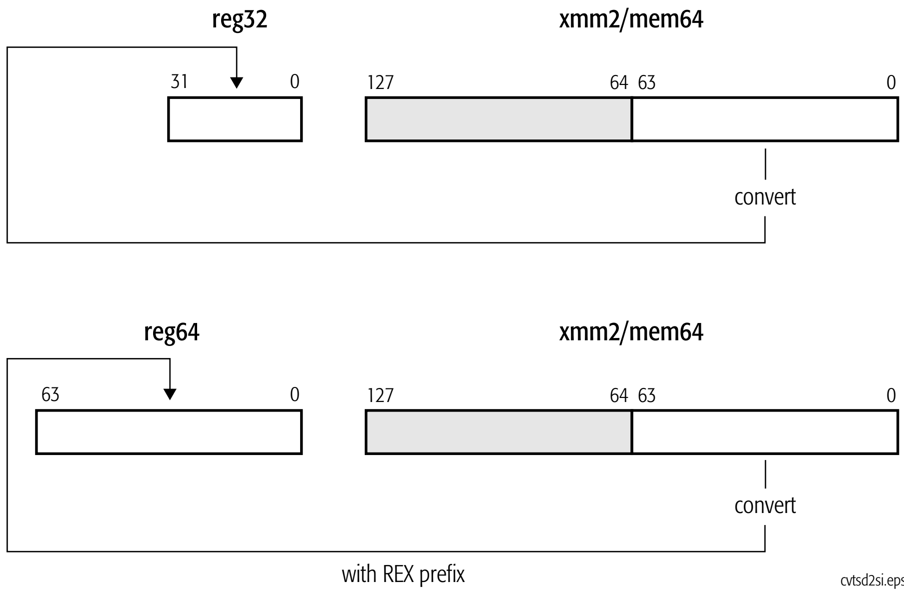
CVTSD2SI Convert Scalar Double-Precision Floating-Point to Signed Doubleword or Quadword Integer

Converts a scalar double-precision floating-point value in the low-order 64 bits of an XMM register or a 64-bit memory location to a 32-bit or 64-bit signed integer and writes the converted value in a general-purpose register.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value (8000_0000h for 32-bit integers, 8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

The CVTSD2SI instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTSD2SI *reg32*, *xmm/mem64*	F2 0F 2D /*r*	Converts a packed double-precision floating-point value in an XMM register or 64-bit memory location to a doubleword integer in a general-purpose register.
CVTSD2SI *reg64*, *xmm/mem64*	F2 0F 2D /*r*	Converts a packed double-precision floating-point value in an XMM register or 64-bit memory location to a quadword integer in a general-purpose register.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

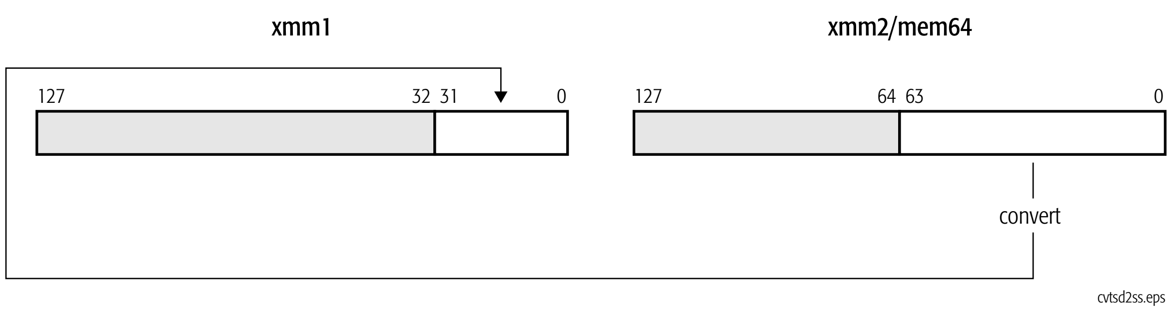
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

CVTSD2SS — Convert Scalar Double-Precision Floating-Point to Scalar Single-Precision Floating-Point

Converts a scalar double-precision floating-point value in the low-order 64 bits of an XMM register or a 64-bit memory location to a single-precision floating-point value and writes the converted value in the low-order 32 bits of another XMM register. The three high-order doublewords in the destination XMM register are not modified. If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

The CVTSD2SS instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTSD2SS *xmm1*, *xmm2/mem64*	F2 0F 5A /*r*	Converts a scalar double-precision floating-point value in an XMM register or 64-bit memory location to a scalar single-precision floating-point value in the destination XMM register.



Related Instructions

CVTPD2PS, CVTPS2PD, CVTSS2SD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

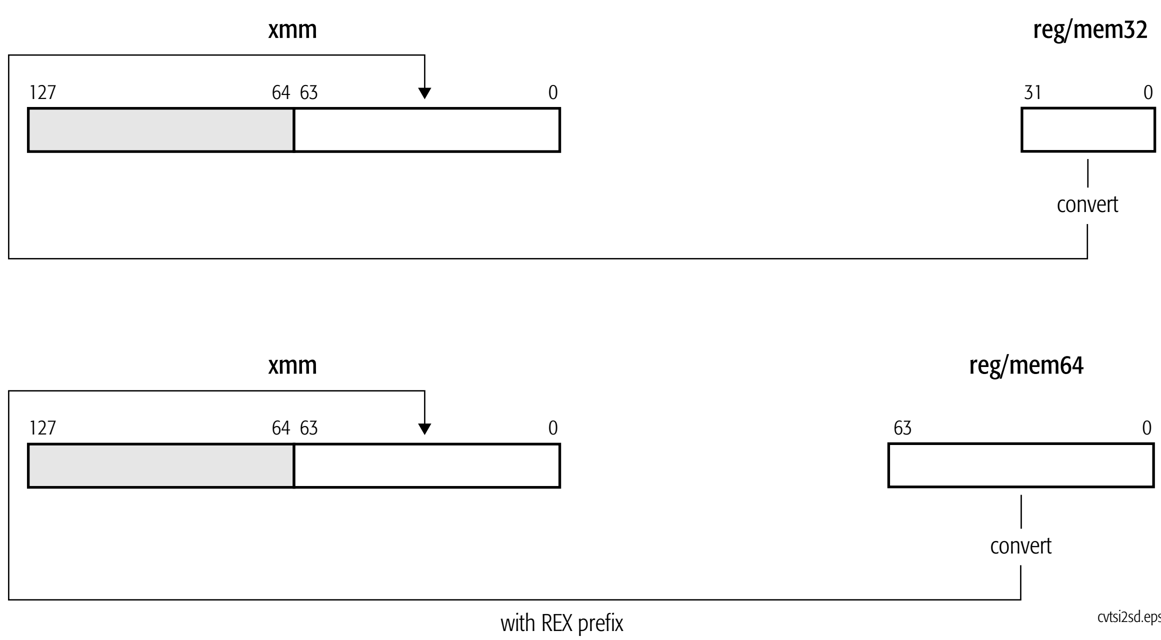
CVTSI2SD Convert Signed Doubleword or Quadword Integer to Scalar Double-Precision Floating-Point

Converts a 32-bit or 64-bit signed integer value in a general-purpose register or memory location to a double-precision floating-point value and writes the converted value in the low-order 64 bits of an XMM register. The high-order 64 bits in the destination XMM register are not modified.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

The CVTSI2SD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTSI2SD *xmm*, *reg/mem32*	F2 0F 2A /*r*	Converts a doubleword integer in a general-purpose register or 32-bit memory location to a double-precision floating-point value in the destination XMM register.
CVTSI2SD *xmm*, *reg/mem64*	F2 0F 2A /*r*	Converts a quadword integer in a general-purpose register or 64-bit memory location to a double-precision floating-point value in the destination XMM register.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTTPD2DQ, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

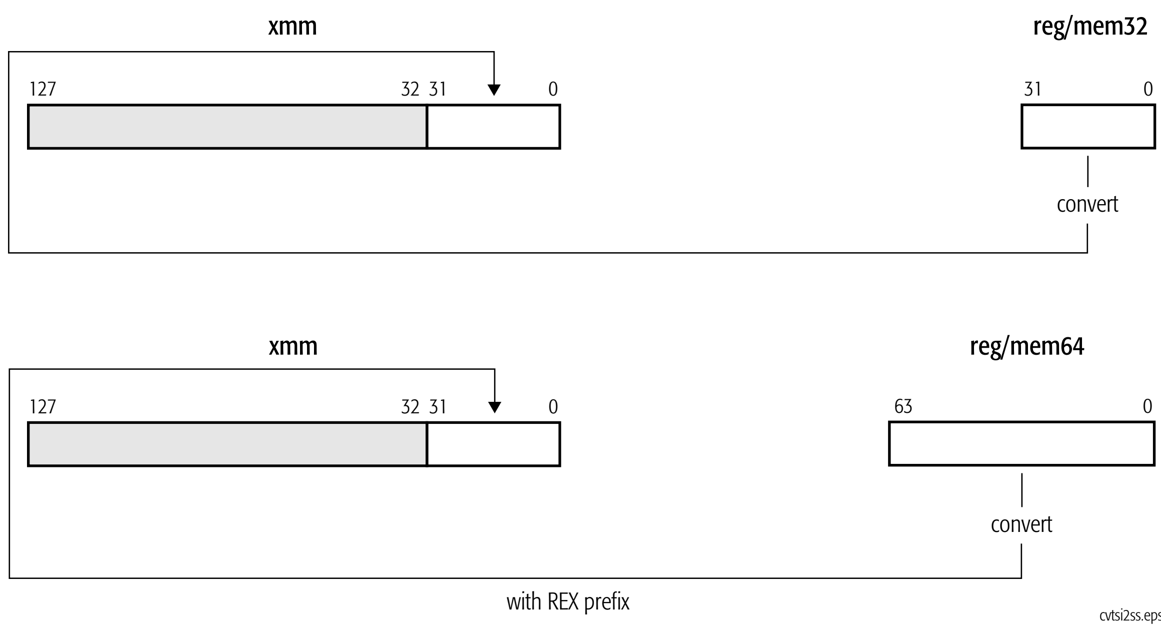
CVTSI2SS Convert Signed Doubleword or Quadword Integer to Scalar Single-Precision Floating-Point

Converts a 32-bit or 64-bit signed integer value in a general-purpose register or memory location to a single-precision floating-point value and writes the converted value in the low-order 32 bits of an XMM register. The three high-order doublewords in the destination XMM register are not modified.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register.

The CVTSI2SS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTSI2SS *xmm*, *reg/mem32*	F3 0F 2A /*r*	Converts a doubleword integer in a general-purpose register or 32-bit memory location to a single-precision floating-point value in the destination XMM register.
CVTSI2SS *xmm*, *reg/mem64*	F3 0F 2A /*r*	Converts a quadword integer in a general-purpose register or 64-bit memory location to a single-precision floating-point value in the destination XMM register.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

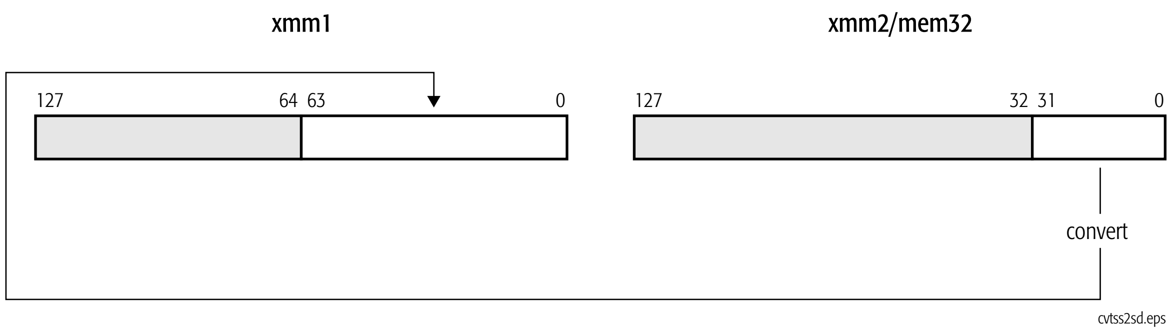
Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

CVTSS2SD Convert Scalar Single-Precision Floating-Point to Scalar Double-Precision Floating-Point

Converts a single-precision floating-point value in the low-order 32 bits of an XMM register or a 32-bit memory location to a double-precision floating-point value and writes the converted value in the low-order 64 bits of another XMM register. The high-order 64 bits in the destination XMM register are not modified.

The CVTSS2SD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTSS2SD *xmm1*, *xmm2/mem32*	F3 0F 5A /*r*	Converts scalar single-precision floating-point value in an XMM register or 32-bit memory location to double-precision floating-point value in the destination XMM register.



Related Instructions

CVTPD2PS, CVTPS2PD, CVTSD2SS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

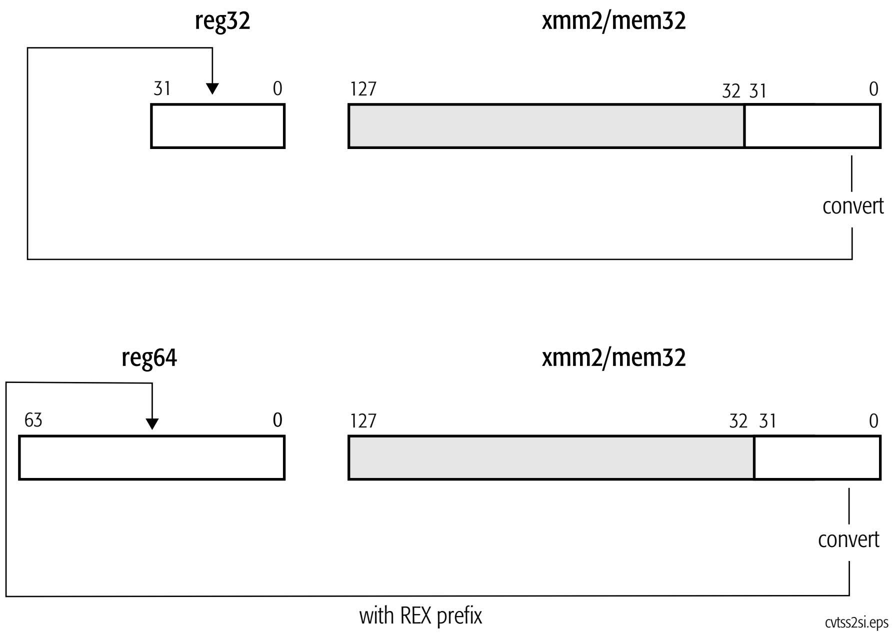
CVTSS2SI Convert Scalar Single-Precision Floating-Point to Signed Doubleword or Quadword Integer

The CVTSS2SI instruction converts a single-precision floating-point value in the low-order 32 bits of an XMM register or a 32-bit memory location to a 32-bit or 64-bit signed integer value and writes the converted value in a general-purpose register.

If the result of the conversion is an inexact value, the value is rounded as specified by the rounding control bits (RC) in the MXCSR register. If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value (8000_0000h for 32-bit integers, 8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

The CVTSS2SI instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTSS2SI *reg32*, *xmm2/mem32*	F3 0F 2D */r*	Converts a single-precision floating-point value in an XMM register or 32-bit memory location to a doubleword integer value in a general-purpose register.
CVTSS2SI *reg64*, *xmm2/mem32*	F3 0F 2D */r*	Converts a single-precision floating-point value in an XMM register or 32-bit memory location to a quadword integer value in a general-purpose register.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTTPS2DQ, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

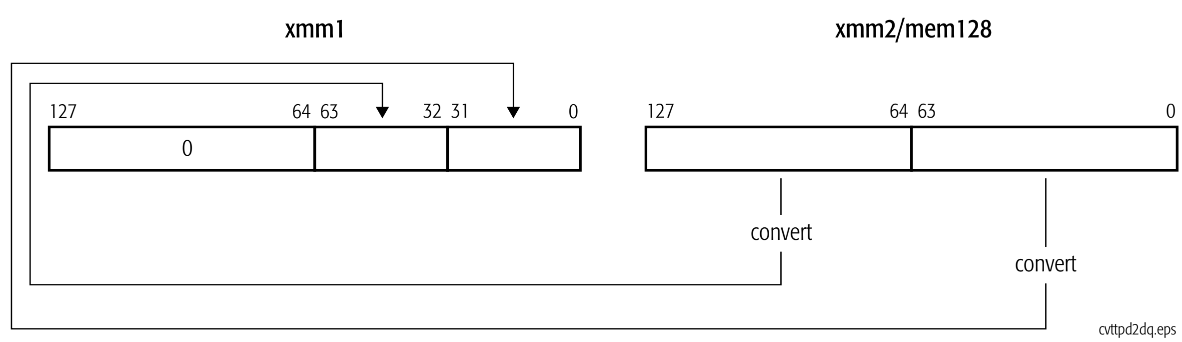
CVTTPD2DQ Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers, Truncated

Converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integer values and writes the converted values in the low-order 64 bits of another XMM register. The high-order 64 bits of the destination XMM register are cleared to all 0s.

If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTTPD2DQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTTPD2DQ *xmm1*, *xmm2/mem128*	66 0F E6 /r	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integer values in the destination XMM register. Inexact results are truncated.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2PI, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

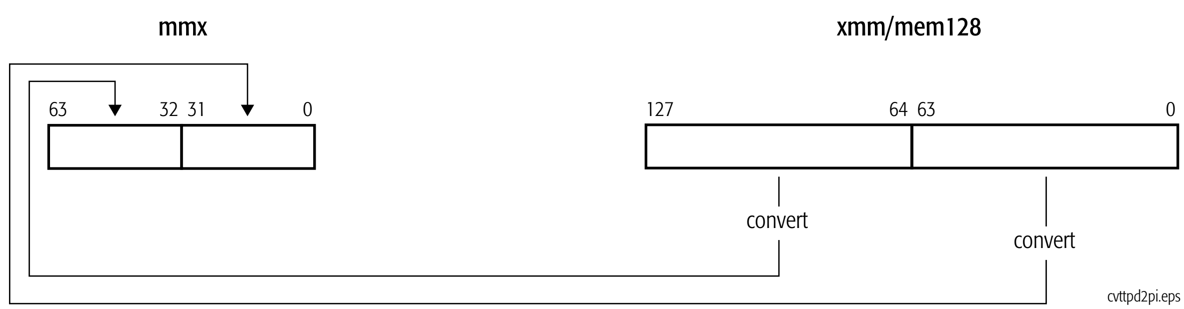
CVTTPD2PI Convert Packed Double-Precision Floating-Point to Packed Doubleword Integers, Truncated

Converts two packed double-precision floating-point values in an XMM register or a 128-bit memory location to two packed 32-bit signed integer values and writes the converted values in an MMX register.

If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTTPD2PI instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTTPD2PI *mmx, xmm/mem128*	66 0F 2C */r*	Converts packed double-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integer values in the destination MMX register. Inexact results are truncated.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTSD2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception is pending due to an x87 floating-point instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

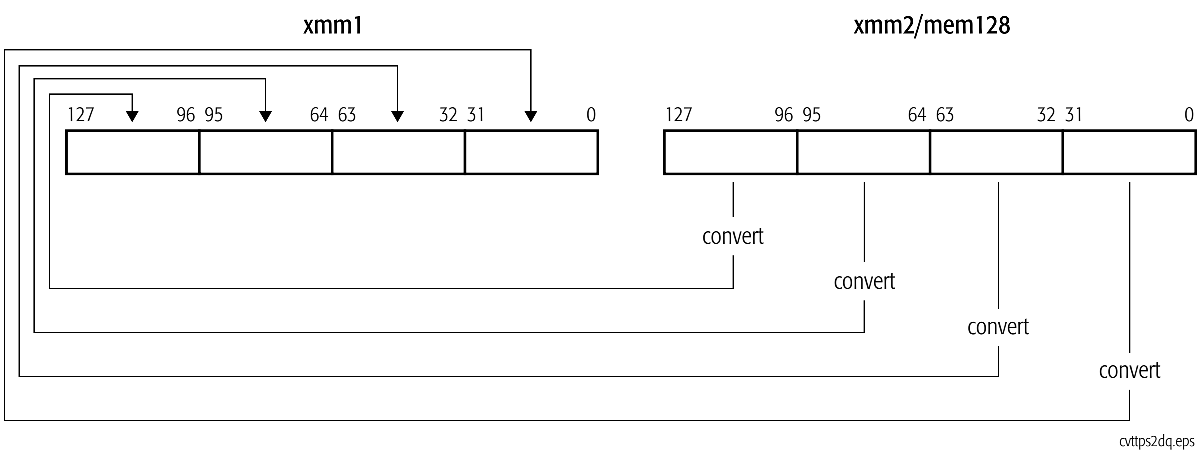
CVTTPS2DQ — Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers, Truncated

Converts four packed single-precision floating-point values in an XMM register or a 128-bit memory location to four packed 32-bit signed integers and writes the converted values in another XMM register.

If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTTPS2DQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
CVTTPS2DQ *xmm1*, *xmm2/mem128*	F3 0F 5B /*r*	Converts packed single-precision floating-point values in an XMM register or 128-bit memory location to packed doubleword integer values in the destination XMM register. Inexact results are truncated.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2PI, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

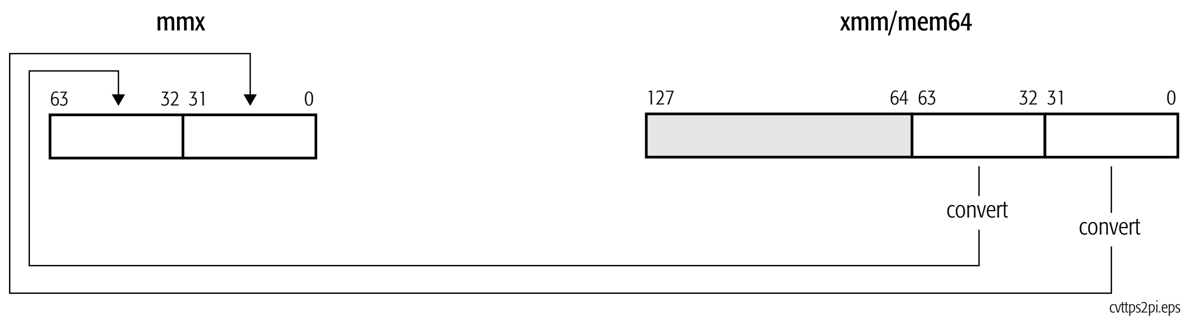
CVTTPS2PI Convert Packed Single-Precision Floating-Point to Packed Doubleword Integers, Truncated

Converts two packed single-precision floating-point values in the low-order 64 bits of an XMM register or a 64-bit memory location to two packed 32-bit signed integer values and writes the converted values in an MMX register.

If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$), the instruction returns the 32-bit indefinite integer value (8000_0000h) when the invalid-operation exception (IE) is masked.

The CVTTPS2PI instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTTPS2PI *mmx, xmm/mem64*	0F 2C /*r*	Converts packed single-precision floating-point values in an XMM register or 64-bit memory location to doubleword integer values in the destination MMX register. Inexact results are truncated.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTSS2SI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An exception was pending due to an x87 floating-point instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.

Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

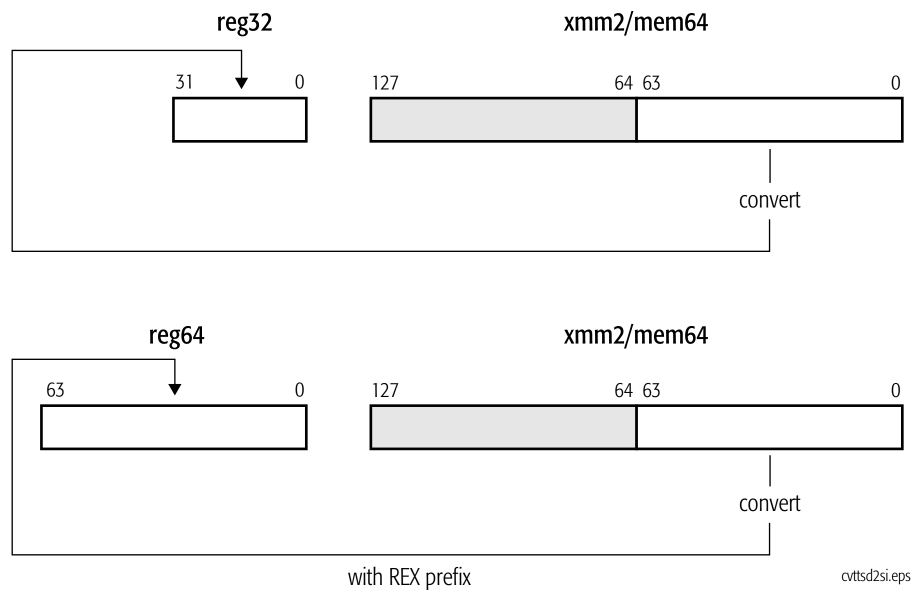
CVTTSD2SI Convert Scalar Double-Precision Floating-Point to Signed Doubleword of Quadword Integer, Truncated

Converts a double-precision floating-point value in the low-order 64 bits of an XMM register or a 64-bit memory location to a 32-bit or 64-bit signed integer value and writes the converted value in a general-purpose register.

If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31}-1$) or quadword value (-2^{63} to $+2^{63}-1$), the instruction returns the indefinite integer value (8000_0000h for 32-bit integers, 8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

The CVTTSD2SI instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTTSD2SI *reg32*, *xmm/mem64*	F2 0F 2C /*r*	Converts scalar double-precision floating-point value in an XMM register or 64-bit memory location to a doubleword signed integer value in a general-purpose register. Inexact results are truncated.
CVTTSD2SI *reg64*, *xmm/mem64*	F2 0F 2C /*r*	Converts scalar double-precision floating-point value in an XMM register or 64-bit memory location to a quadword signed integer value in a general-purpose register. Inexact results are truncated.



Related Instructions

CVTDQ2PD, CVTPD2DQ, CVTPD2PI, CVTPI2PD, CVTSD2SI, CVTSI2SD, CVTTPD2DQ, CVTTPD2PI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

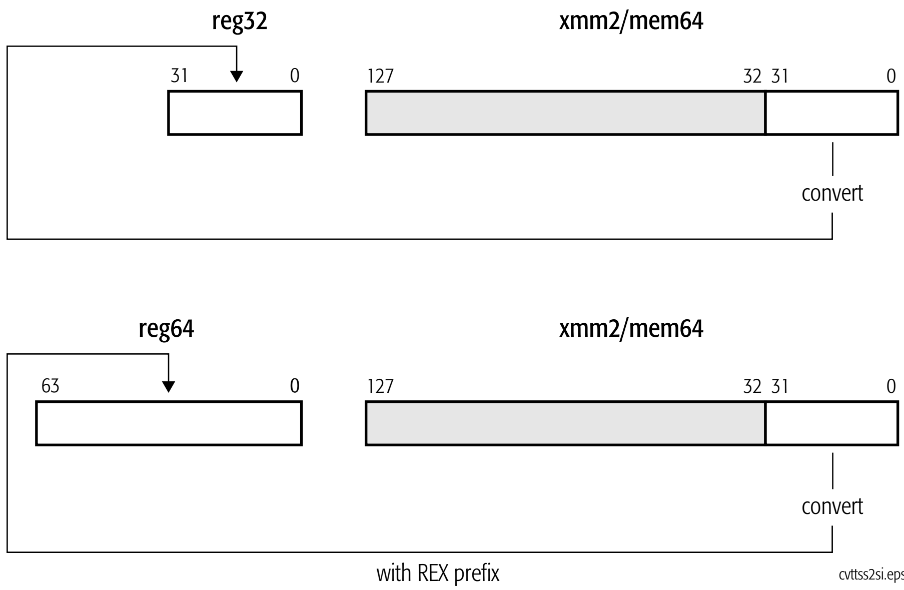
CVTTSS2SI Convert Scalar Single-Precision Floating-Point to Signed Doubleword or Quadword Integer, Truncated

Converts a single-precision floating-point value in the low-order 32 bits of an XMM register or a 32-bit memory location to a 32-bit or 64-bit signed integer value and writes the converted value in a general-purpose register.

If the result of the conversion is an inexact value, the value is truncated (rounded toward zero). If the floating-point value is a NaN, infinity, or if the result of the conversion is larger than the maximum signed doubleword (-2^{31} to $+2^{31} - 1$) or quadword value (-2^{63} to $+2^{63} - 1$), the instruction returns the indefinite integer value (8000_0000h for 32-bit integers, 8000_0000_0000_0000h for 64-bit integers) when the invalid-operation exception (IE) is masked.

The CVTTSS2SI instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
CVTTSS2SI *reg32*, *xmm/mem32*	F3 0F 2C */r*	Converts scalar single-precision floating-point value in an XMM register or 32-bit memory location to a signed doubleword integer value in a general-purpose register. Inexact results are truncated.
CVTTSS2SI *reg64*, *xmm/mem32*	F3 0F 2C */r*	Converts scalar single-precision floating-point value in an XMM register or 32-bit memory location to a signed quadword integer value in a general-purpose register. Inexact results are truncated.



Related Instructions

CVTDQ2PS, CVTPI2PS, CVTPS2DQ, CVTPS2PI, CVTSI2SS, CVTSS2SI, CVTTPS2DQ, CVTTPS2PI

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M					M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

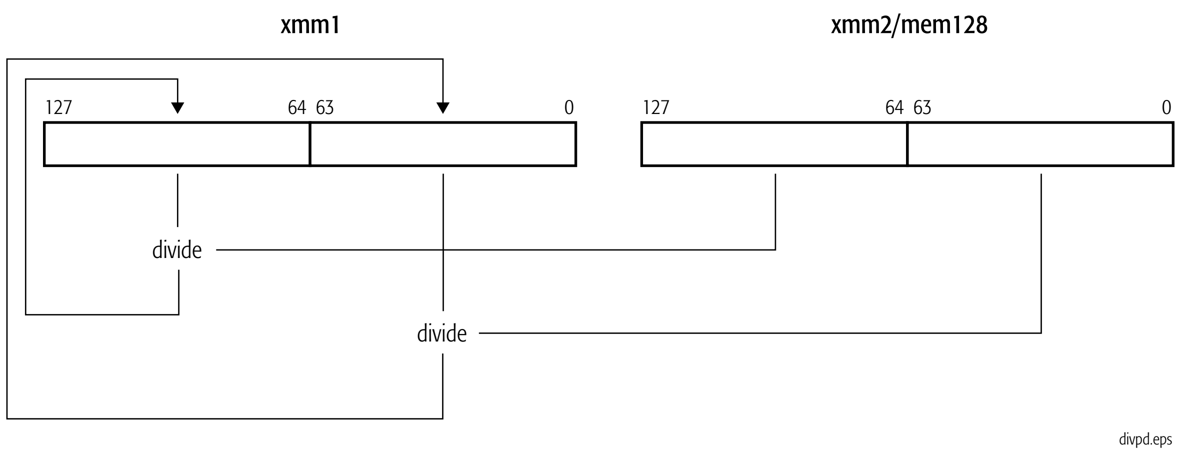
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value, a QNaN value, or ±infinity.
	X	X	X	A source operand was too large to fit in the destination format.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

DIVPD Divide Packed Double-Precision Floating-Point

Divides each of the two packed double-precision floating-point values in the first source operand by the corresponding packed double-precision floating-point value in the second source operand and writes the result of each division in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The DIVPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
DIVPD *xmm1*, *xmm2/mem128*	66 0F 5E /*r*	Divides packed double-precision floating-point values in an XMM register by the packed double-precision floating-point values in another XMM register or 128-bit memory location.



Related Instructions

DIVPS, DIVSD, DIVSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M	M	M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was divided by ±zero.
	X	X	X	±infinity was divided by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

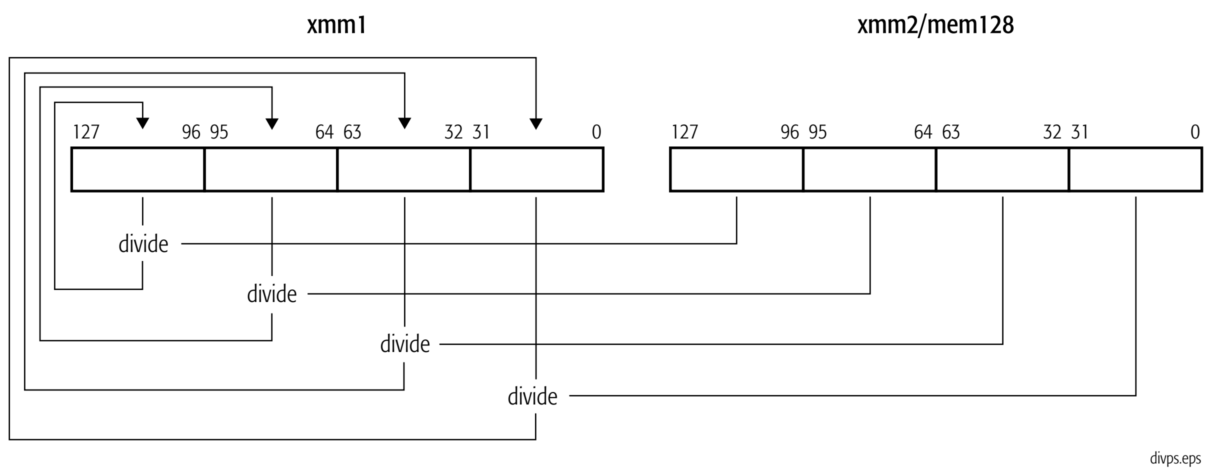
Exception	Real	Virtual 8086	Protected	Cause of Exception
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Zero-divide exception (ZE)	X	X	X	A non-zero number was divided by zero.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

DIVPS Divide Packed Single-Precision Floating-Point

Divides each of the four packed single-precision floating-point values in the first source operand by the corresponding packed single-precision floating-point value in the second source operand and writes the result of each division in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The DIVPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
DIVPS *xmm1*, *xmm/2mem128*	0F 5E */r*	Divides packed single-precision floating-point values in an XMM register by the packed single-precision floating-point values in another XMM register or 128-bit memory location.



Related Instructions

DIVPD, DIVSD, DIVSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M	M	M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was divided by ±zero.
	X	X	X	±infinity was divided by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

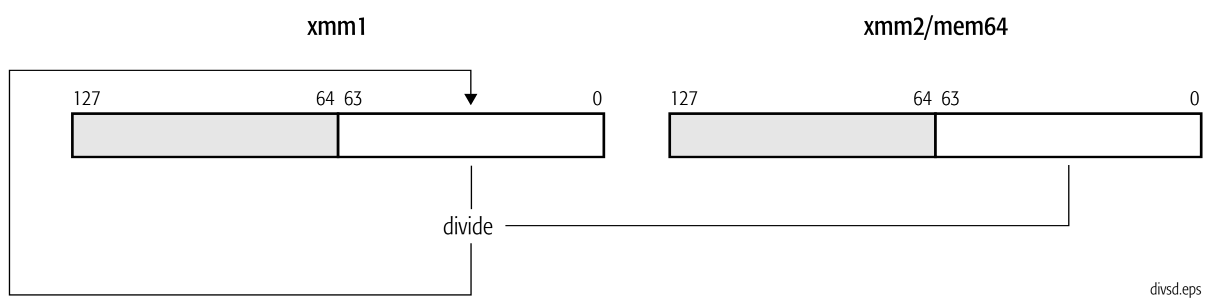
Exception	Real	Virtual 8086	Protected	Cause of Exception
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Zero-divide exception (ZE)	X	X	X	A non-zero number was divided by zero.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

DIVSD Divide Scalar Double-Precision Floating-Point

Divides the double-precision floating-point value in the low-order quadword of the first source operand by the double-precision floating-point value in the low-order quadword of the second source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The DIVSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
DIVSD *xmm1*, *xmm2/mem64*	F2 0F 5E /*r*	Divides low-order double-precision floating-point value in an XMM register by the low-order double-precision floating-point value in another XMM register or in a 64- or 128-bit memory location.



Related Instructions

DIVPD, DIVPS, DIVSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M	M	M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was divided by ±zero.
	X	X	X	±infinity was divided by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

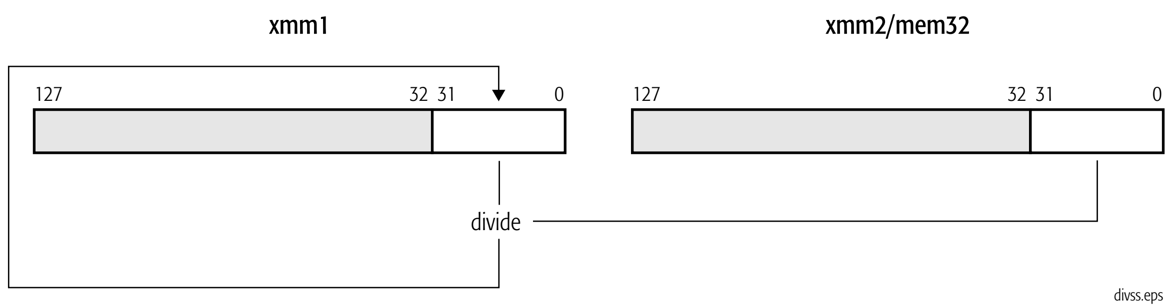
Exception	Real	Virtual 8086	Protected	Cause of Exception
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Zero-divide exception (ZE)	X	X	X	A non-zero number was divided by zero.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

DIVSS Divide Scalar Single-Precision Floating-Point

Divides the single-precision floating-point value in the low-order doubleword of the first source operand by the single-precision floating-point value in the low-order doubleword of the second source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The DIVSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
DIVSS *xmm1*, *xmm2/mem32*	F3 0F 5E /*r*	Divides low-order single-precision floating-point value in an XMM register by the low-order single-precision floating-point value in another XMM register or in a 32-bit memory location.



Related Instructions

DIVPD, DIVPS, DIVSD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M	M	M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was divided by ±zero.
	X	X	X	±infinity was divided by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Zero-divide exception (ZE)	X	X	X	A non-zero number was divided by zero.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

FXRSTOR Restore XMM, MMX, and x87 State

Restores the XMM, MMX, and x87 state. The data loaded from memory is the state information previously saved using the FXSAVE instruction. Restoring data with FXRSTOR that had been previously saved with an FSAVE (rather than FXSAVE) instruction results in an incorrect restoration.

If FXRSTOR results in set exception flags in the loaded x87 status word register, and these exceptions are unmasked in the x87 control word register, a floating-point exception occurs when the next floating-point instruction is executed (except for the no-wait floating-point instructions).

If the restored MXCSR register contains a set bit in an exception status flag, and the corresponding exception mask bit is cleared (indicating an unmasked exception), loading the MXCSR register from memory does not cause a SIMD floating-point exception (#XF).

FXRSTOR does not restore the x87 error pointers (last instruction pointer, last data pointer, and last opcode), except in the relatively rare cases in which the exception-summary (ES) bit in the x87 status word is set to 1, indicating that an unmasked x87 exception has occurred.

The architecture supports two 512-bit memory formats for FXRSTOR, a 64-bit format that loads XMM0-XMM15, and a 32-bit legacy format that loads only XMM0-XMM7. If FXRSTOR is executed in 64-bit mode, the 64-bit format is used, otherwise the 32-bit format is used. When the 64-bit format is used, if the operand-size is 64-bit, FXRSTOR loads the x87 pointer registers as *offset64*, otherwise it loads them as *sel:offset32*. For details about the memory format used by FXRSTOR, see "Saving Media and x87 Processor State" in volume 2.

If the fast-FXSAVE/FXRSTOR (FFXSR) feature is enabled in EFER, FXRSTOR does not restore the XMM registers (XMM0-XMM15) when executed in 64-bit mode at CPL 0. MXCSR is restored whether fast-FXSAVE/FXRSTOR is enabled or not. Software can use CPUID to determine whether the fast-FXSAVE/FXRSTOR feature is available. (See “CPUID” in Volume 3.)

If the operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0, the saved image of XMM0–XMM15 and MXCSR is not loaded into the processor. A general-protection exception occurs if there is an attempt to load a non-zero value to the bits in MXCSR that are defined as reserved (bits 31–16).

.

Mnemonic	Opcode	Description
FXRSTOR *mem512env*	0F AE /1	Restores XMM, MM™, and x87 state from 512-byte memory location.

Related Instructions

FWAIT, FXSAVE

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The FXSAVE/FXRSTOR instructions are not supported, as indicated by EDX bit 24 of CPUID standard funcion 1 or extended function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
	X	X	X	Ones were written to the reserved bits in MXCSR.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

FXSAVE — Save XMM, MMX, and x87 State

Saves the XMM, MMX, and x87 state. A memory location that is not aligned on a 16-byte boundary causes a general-protection exception.

Unlike FSAVE and FNSAVE, FXSAVE does not alter the x87 tag bits. The contents of the saved MMX/x87 data registers are retained, thus indicating that the registers may be valid (or whatever other value the x87 tag bits indicated prior to the save). To invalidate the contents of the MMX/x87 data registers after FXSAVE, software must execute an FINIT instruction. Also, FXSAVE (like FNSAVE) does not check for pending unmasked x87 floating-point exceptions. An FWAIT instruction can be used for this purpose.

FXSAVE does not save the x87 pointer registers (last instruction pointer, last data pointer, and last opcode), except in the relatively rare cases in which the exception-summary (ES) bit in the x87 status word is set to 1, indicating that an unmasked x87 exception has occurred.

The architecture supports two 512-bit memory formats for FXSAVE, a 64-bit format that saves XMM0-XMM15, and a 32-bit legacy format that saves only XMM0-XMM7. If FXSAVE is executed in 64-bit mode, the 64-bit format is used, otherwise the 32-bit format is used. When the 64-bit format is used, if the operand-size is 64-bit, FXSAVE saves the x87 pointer registers as *offset64*, otherwise it saves them as *sel:offset32*. For more details about the memory format used by FXSAVE, see "Saving Media and x87 Processor State" in volume 2.

If the fast-FXSAVE/FXRSTOR (FFXSR) feature is enabled in EFER, FXSAVE does not save the XMM registers (XMM0-XMM15) when executed in 64-bit mode at CPL 0. MXCSR is saved whether fast-FXSAVE/FXRSTOR is enabled or not. Software can use CPUID to determine whether the fast-FXSAVE/FXRSTOR feature is available. (See "CPUID" in Volume 3.)

If the operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0, FXSAVE does not save the image of XMM0–XMM15 or MXCSR. For details about the CR4.OSFXSR bit, see "FXSAVE/FXRSTOR Support (OSFXSR) Bit" in Volume 2.

Mnemonic	Opcode	Description
FXSAVE *mem512env*	0F AE /0	Saves XMM, MMX, and x87 state to 512-byte memory location.

Related Instructions

FINIT, FNSAVE, FRSTOR, FSAVE, FXRSTOR, LDMXCSR, STMXCSR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

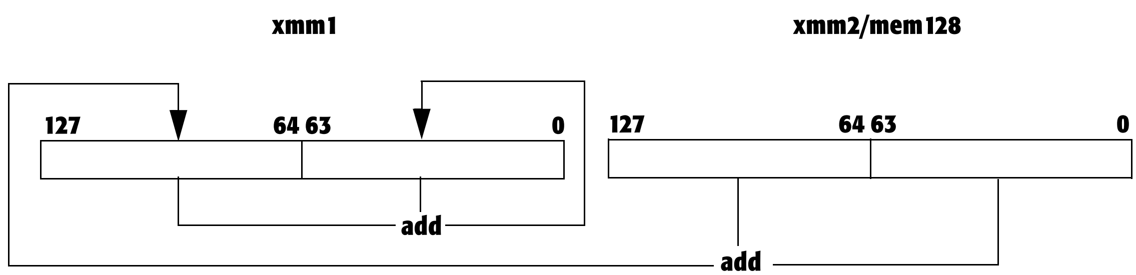
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The FXSAVE/FXRSTOR instructions are not supported, as indicated by EDX bit 24 of CPUID standard function 1 or extended function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

HADDPD Horizontal Add Packed Double

Adds the double-precision floating-point values in the high and low quadwords of the destination operand and stores the result in the low quadword of the destination operand. Simultaneously, the instruction adds the double-precision floating-point values in the high and low quadwords of the source operand and stores the result in the high quadword of the destination operand.

The HADDPD instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
HADDPD *xmm1*, *xmm2/mem128*	66 0F 7C /r	Adds two packed double-precision values in *xmm1* and stores the result in the lower 64 bits of *xmm1*; adds two packed double-precision values in *xmm2* or a 128-bit memory operand and stores the result in the upper 64 bits of *xmm1*.



Related Instructions

HADDPS, HSUBPD, HSUBPS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

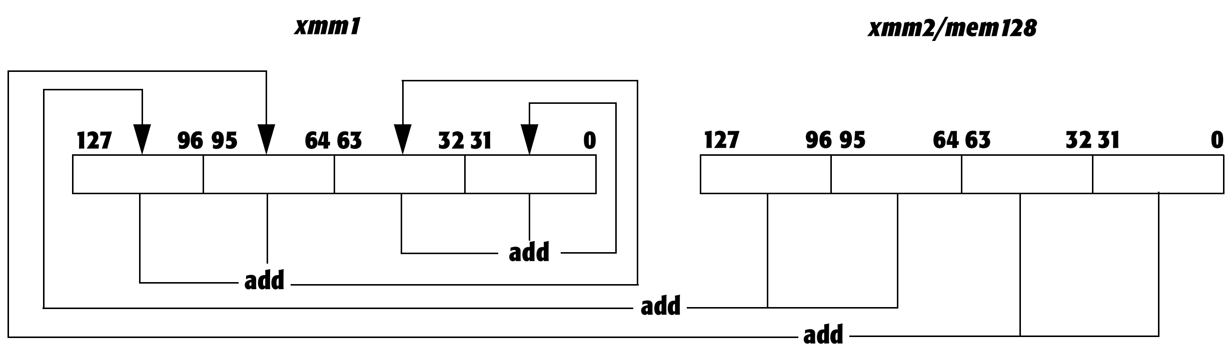
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS			X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

HADDPS Horizontal Add Packed Single

Adds pairs of packed single-precision floating-point values simultaneously. The sum of the values in the first and second doublewords of the destination operand is stored in the first doubleword of the destination operand; the sum of the values in the third and fourth doubleword of the destination operand is stored in the second doubleword of the destination operand; the sum of the values in the first and second doubleword of the source operand is stored in the third doubleword of the destination operand; and the sum of the values in the third and fourth doubleword of the source operand is stored in the fourth doubleword of the destination operand.

The HADDPS instruction is an SSE3 instruction;. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
HADDPS *xmm1*, *xmm2/mem128*	F2 0F 7C /r	Adds the first an second packed single-precision values in xmm1 and stores the sum in xmm1[0-31]; adds the third and fourth single-precision values in xmm1 and stores the sum in xmm1[32–63]; adds the first and second packed single-precision values in xmm2 or a 128-bit memory operand and stores the sum in the xmm1[64–95]; adds the third and fourth packed single-precision values in xmm2 or a 128-bit memory operand and stores the result in xmm1[96–127].



Related Instructions

HADDPD, HSUBPD, HSUBPS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS			X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.

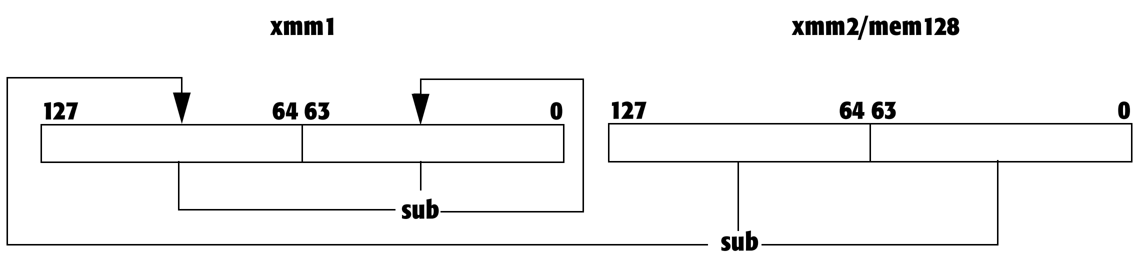
Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was added to –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

HSUBPD Horizontal Subtract Packed Double

Subtracts the packed double-precision floating-point value in the upper quadword of the destination XMM register operand from the lower quadword of the destination operand and stores the result in the lower quadword of the destination operand; subtracts the value in the upper quadword of the source XMM register or 128-bit memory operand from the value in the lower quadword of the source operand and stores the result in the upper quadword of the destination XMM register.

The HSUBPD instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
HSUBPD *xmm1*, *xmm2/mem128*	66 0F 7D /r	Subtracts the packed double-precision value in the upper 64 bits of the source register from the value in the lower 64 bits of the source register or 128-bit memory operand and stores the difference in the upper 64 bits of the destination XMM register; Subtracts the upper 64 bits of the destination register from the lower 64 bits of the destination register and stores the result in the lower 64 bits of the destination XMM register.



Related Instructions

HSUBPS, HADDPD, HADDPS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS			X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.

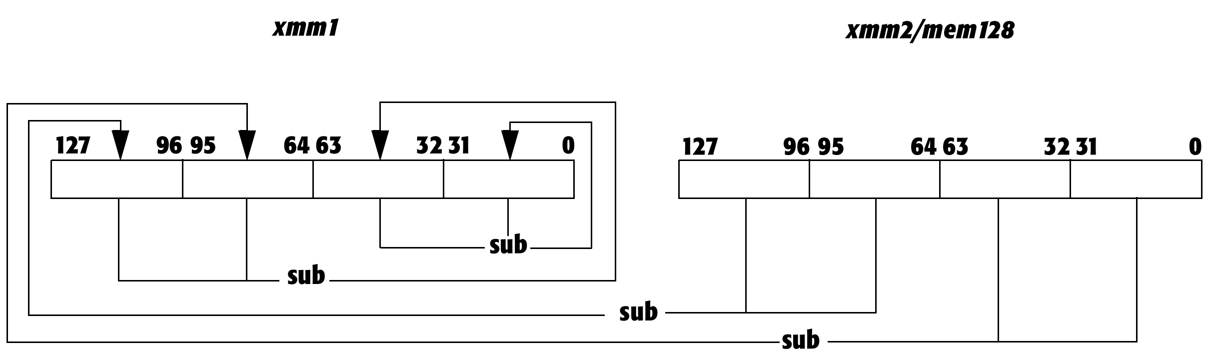
Exception	Real	Virtual 8086	Protected	Cause of Exception
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

HSUBPS Horizontal Subtract Packed Single

Subtracts the packed single-precision floating-point value in the second doubleword of the destination XMM register from that in the first doubleword of the destination register and stores the difference in the first doubleword of the destination register; subtracts the value in the fourth doubleword of the destination register from that in the third doubleword of the destination register and stores the result in the second doubleword of the destination register; subtracts the value in the second doubleword of the source XMM register or 128-bit memory operand from the first doubleword of the source operand and stores the result in the third doubleword of the destination XMM register; subtracts the single-precision floating-point value in the fourth doubleword of the source operand from the third doubleword of the source operand and stores the result in the fourth doubleword of the destination XMM register.

The HSUBPS instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
HSUBPS *xmm1*, *xmm2/mem128*	F2 0F 7D /r	Subtracts the second 32 bits of the destination operand from the first 32 bits of the destination operand and stores the difference in the first first doubleword of the destination operand; subtracts the fourth 32 bits of the destination operand from the third 32-bits of the destination operand and and stores the difference in the second doubleword of the destination operand; subtracts the second 32 bits of the source operand from the first 32 bits of the source operand and stores the difference in the third doubleword of the destination operand; subtracts the fourth 32-bits of the source operand from the third 32 bits of the source operand and stores the difference in the fourth doubleword of of the destination operand.



Related Instructions

HSUBPD, HADDPD, HADDPS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS			X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* below for details.
SIMD Floating-Point Exceptions				

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

LDDQU Load Unaligned Double Quadword

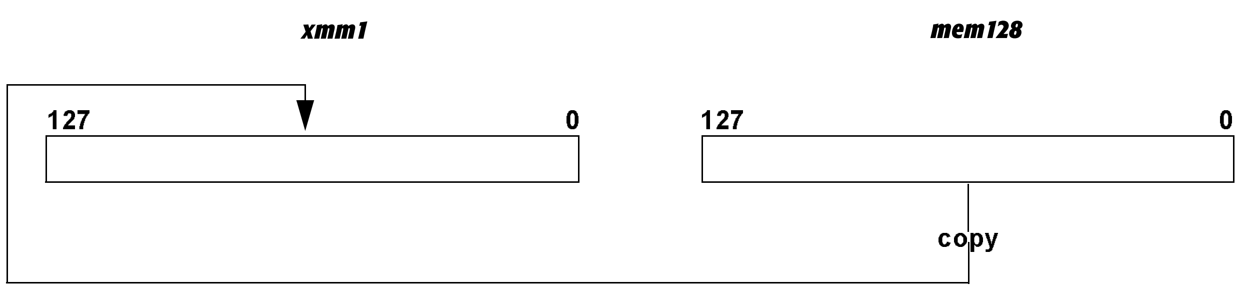
Moves an unaligned 128-bit (double quadword) value from a 128-bit memory location to a destination XMM register.

Like the MOVUPD instruction, the LDDQU instruction loads a 128-bit operand from an unaligned memory location. However, to improve performance when the memory operand is actually misaligned, LDDQU may read an aligned 16 bytes to get the first part of the operand, and an aligned 16 bytes to get the second part of the operand. This behavior is implementation-specific, and LDDQU may only read the exact 16 bytes needed for the memory operand. If the memory operand is in a memory range where reading extra bytes can cause performance or functional issues, use the MOVUPD instruction instead of LDDQU.

Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

The LDDQU instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
LDDQU *xmm1*, *mem128*	F2 0F F0 /*r*	Moves a 128-bit value from an unaligned 128-bit memory location to the destination XMM register.



Related Instructions

MOVDQU

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indcated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned-memory reference was performed while alignment checking was enabled.

LDMXCSR — Load MXCSR Control/Status Register

Loads the MXCSR register with a 32-bit value from memory. The least-significant bit of the memory location is loaded in bit 0 of MXCSR. Bits 31–16 of the MXCSR are reserved and must be zero. A general-protection exception occurs if the LDMXCSR instruction attempts to load non-zero values into MXCSR bits 31–16.

The MXCSR register is described in “Registers” in Volume 1.

The LDMXCSR instruction is an SSE instruction; check the status of EDX bit 25 returned by CPUID standard function 1 to verify that the processor supports this function. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
LDMXCSR *mem32*	0F AE /2	Loads MXCSR register with 32-bit value in memory.

Related Instructions

STMXCSR

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	Ones were written to the reserved bits in MXCSR.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MASKMOVDQU Masked Move Double Quadword Unaligned

Stores bytes from the first source operand as selected by the sign bits in the second source operand (sign-bit is 0 = no write and sign-bit is 1 = write) to a memory location specified in the DS:rDI registers. The first source operand is an XMM register, and the second source operand is another XMM register. The store address may be unaligned.

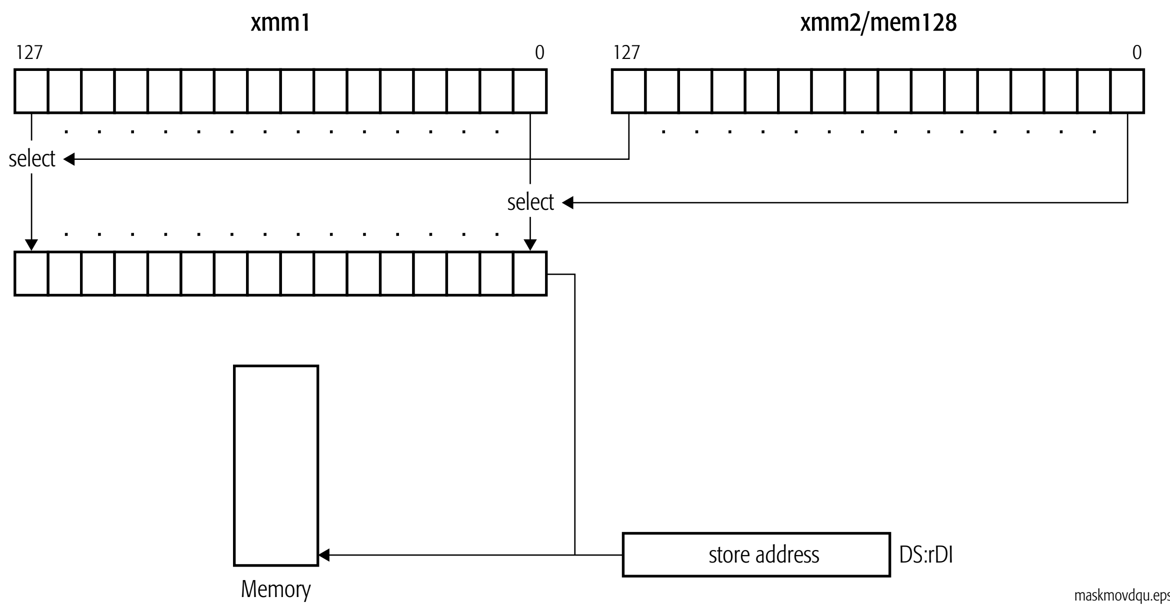
A mask value of all 0s results in the following behavior:

- No data is written to memory.
- Code and data breakpoints are not guaranteed to be signaled in all implementations.
- Exceptions associated with memory addressing and page faults are not guaranteed to be signaled in all implementations.

MASKMOVDQU implicitly uses weakly-ordered, write-combining buffering for the data, as described in "Buffering and Combining Memory Writes" in Volume 2. For data that is shared by multiple processors, this instruction should be used together with a fence instruction in order to ensure data coherency (refer to "Cache and TLB Management" in Volume 2).

The MASKMOVDQU instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MASKMOVDQU *xmm*1, *xmm*2	66 0F F7 /*r*	Store bytes from an XMM register selected by a mask value in another XMM register to DS:rDI.



Related Instructions

MASKMOVQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.

Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

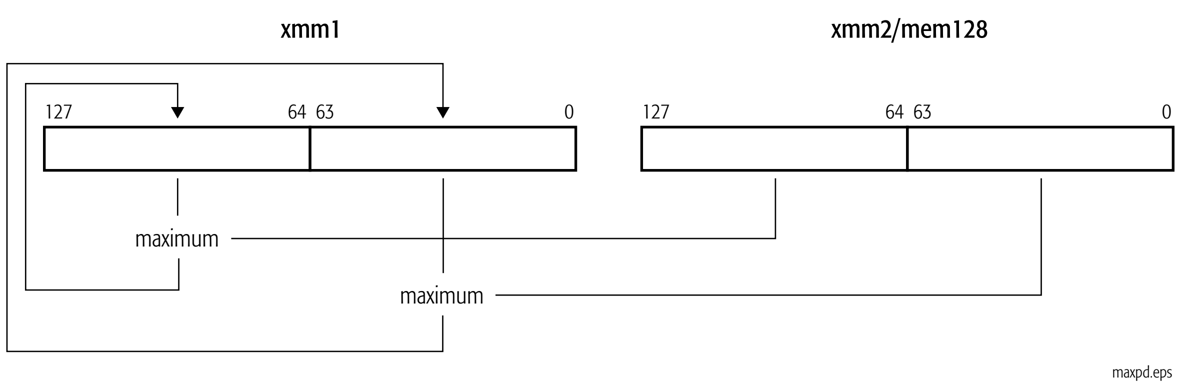
MAXPD Maximum Packed Double-Precision Floating-Point

Compares each of the two packed double-precision floating-point values in the first source operand with the corresponding packed double-precision floating-point value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MAXPD instructionn is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MAXPD *xmm1*, *xmm2/mem128*	66 0F 5F /*r*	Compares two pairs of packed double-precision values in an XMM register and another XMM register or 128-bit memory location and writes the greater value of each comparison in the destination XMM register.



Related Instructions

MAXPS, MAXSD, MAXSS, MINPD, MINPS, MINSD, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

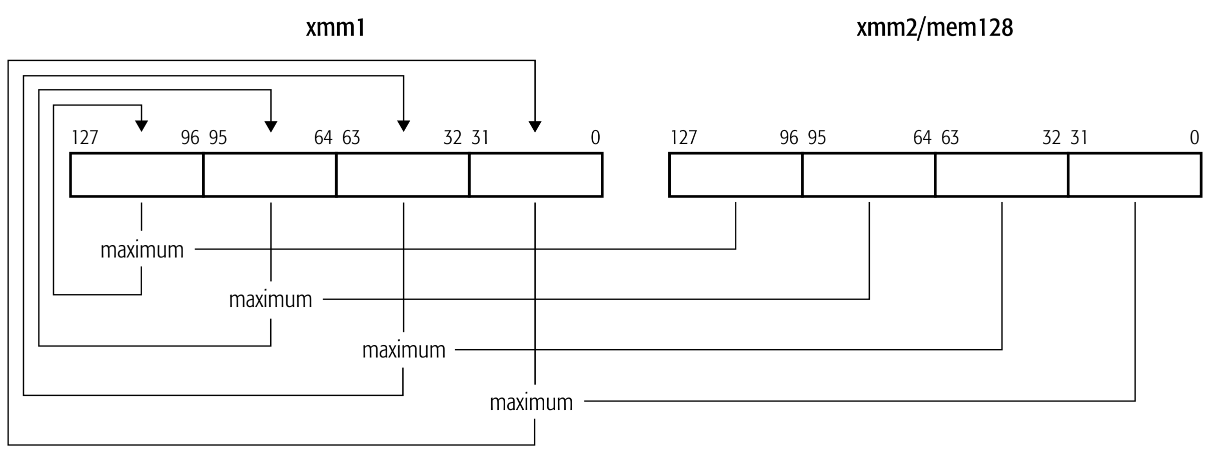
MAXPS Maximum Packed Single-Precision Floating- Point

Compares each of the four packed single-precision floating-point values in the first source operand with the corresponding packed single-precision floating-point value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MAXPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MAXPS *xmm1*, *xmm2/mem128*	0F 5F */r*	Compares four pairs of packed single-precision values in an XMM register and another XMM register or 128-bit memory location and writes the maximum value of each comparison in the destination XMM register.



Related Instructions

MAXPD, MAXSD, MAXSS, MINPD, MINPS, MINSD, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

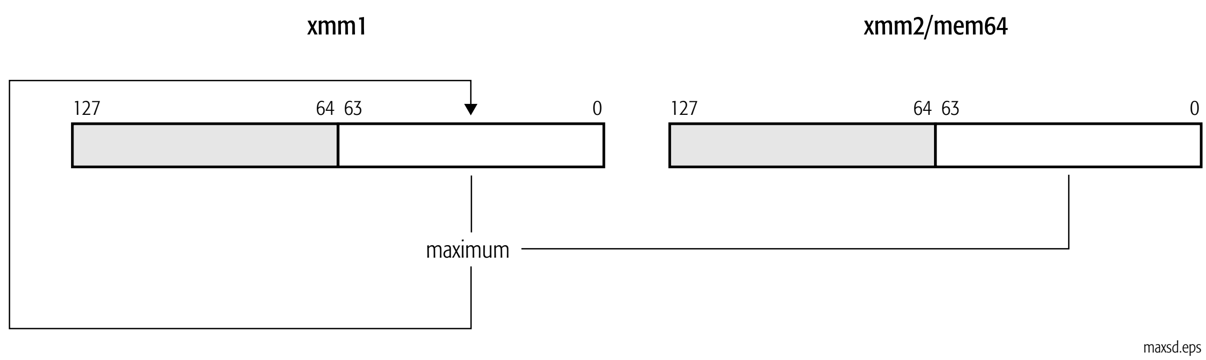
MAXSD Maximum Scalar Double-Precision Floating- Point

Compares the double-precision floating-point value in the low-order 64 bits of the first source operand with the double-precision floating-point value in the low-order 64 bits of the second source operand and writes the numerically greater of the two values in the low-order quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 64-bit memory location. The high-order quadword of the destination XMM register is not modified.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MAXSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MAXSD *xmm1*, *xmm2/mem64*	F2 0F 5F /*r*	Compares scalar double-precision values in an XMM register and another XMM register or 64-bit memory location and writes the greater of the two values in the destination XMM register.



Related Instructions

MAXPD, MAXPS, MAXSS, MINPD, MINPS, MINSD, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

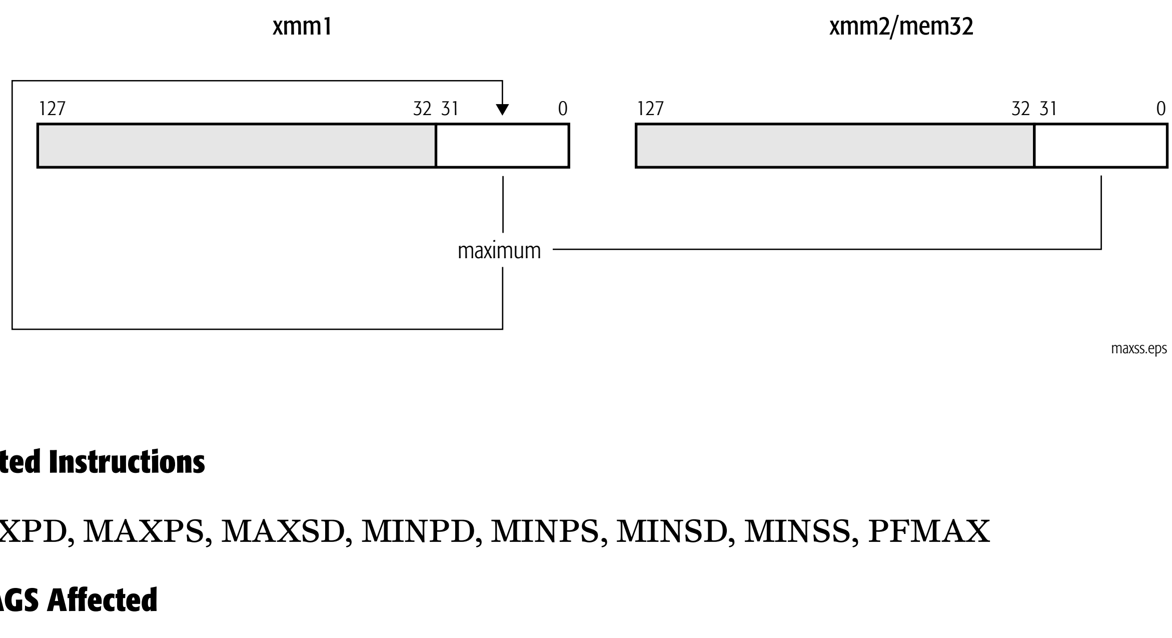
MAXSS Maximum Scalar Single-Precision Floating- Point

Compares the single-precision floating-point value in the low-order 32 bits of the first source operand with the single-precision floating-point value in the low-order 32 bits of the second source operand and writes the numerically greater of the two values in the low-order 32 bits of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 32-bit memory location. The three high-order doublewords of the destination XMM register are not modified.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MAXSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MAXSS *xmm1*, *xmm2/mem32*	F3 0F 5F */r*	Compares scalar single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the greater of the two values in the destination XMM register.



Related Instructions

MAXPD, MAXPS, MAXSD, MINPD, MINPS, MINSD, MINSS, PFMAX

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

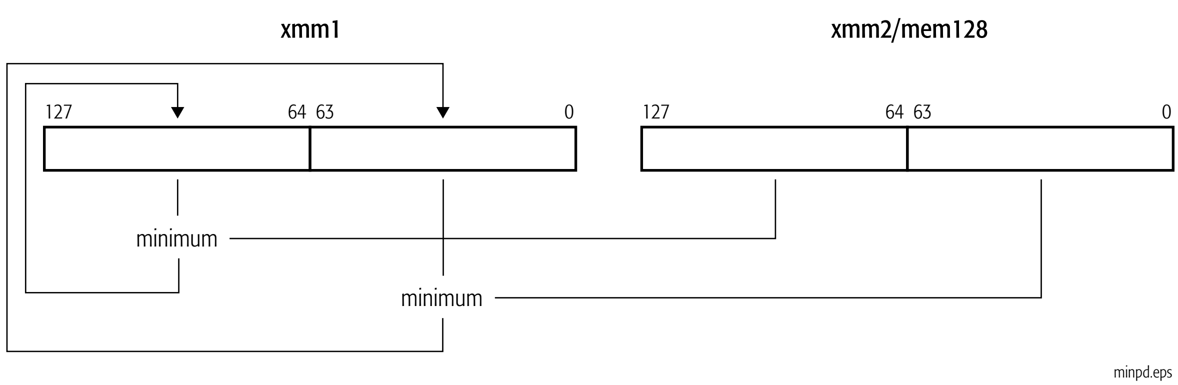
MINPD Minimum Packed Double-Precision Floating- Point

Compares each of the two packed double-precision floating-point values in the first source operand with the corresponding packed double-precision floating-point value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 128-bit memory location.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MINPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MINPD *xmm1*, *xmm2/mem128*	66 0F 5D */r*	Compares two pairs of packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the lesser value of each comparison in the destination XMM register.



Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPS, MINSD, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions* for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

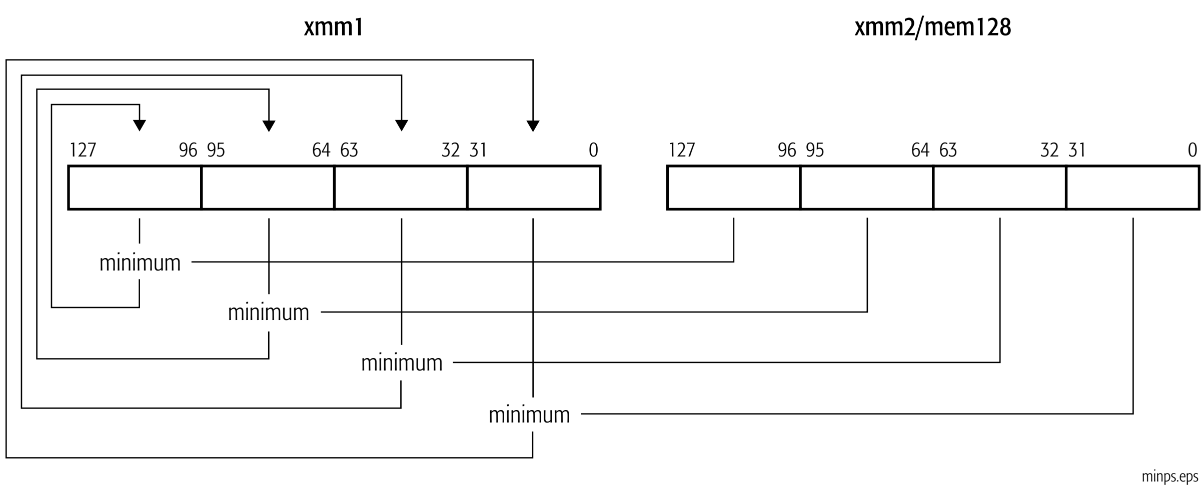
MINPS Minimum Packed Single-Precision Floating-Point

The MINPS instruction compares each of the four packed single-precision floating-point values in the first source operand with the corresponding packed single-precision floating-point value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 128-bit memory location.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MINPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MINPS *xmm1*, *xmm2/mem128*	0F 5D /*r*	Compares four pairs of packed single-precision values in an XMM register and another XMM register or 128-bit memory location and writes the numerically lesser value of each comparison in the destination XMM register.



Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPD, MINSD, MINSS, PFMIN

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

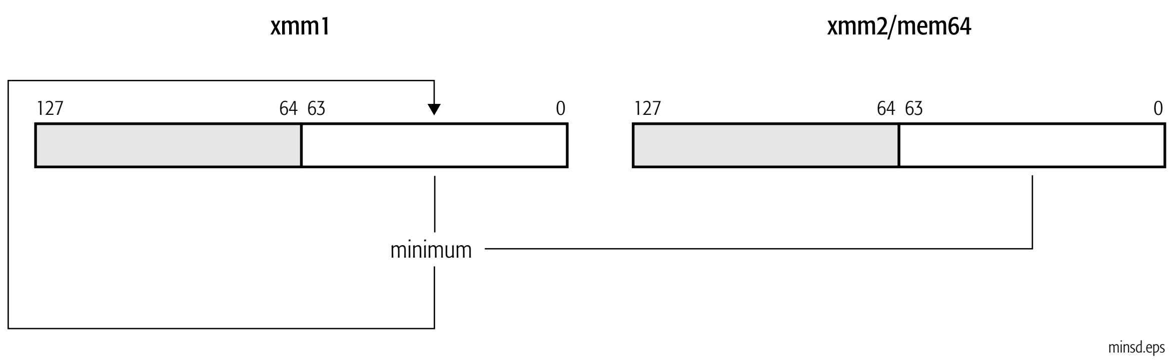
MINSD Minimum Scalar Double-Precision Floating- Point

Compares the double-precision floating-point value in the low-order 64 bits of the first source operand with the double-precision floating-point value in the low-order 64 bits of the second source operand and writes the numerically lesser of the two values in the low-order 64 bits of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 64-bit memory location. The high-order quadword of the destination XMM register is not modified.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MINSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MINSD *xmm1*, *xmm2/mem64*	F2 0F 5D */r*	Compares scalar double-precision floating-point values in an XMM register and another XMM register or 64-bit memory location and writes the lesser of the two values in the destination XMM register.



Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPD, MINPS, MINSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

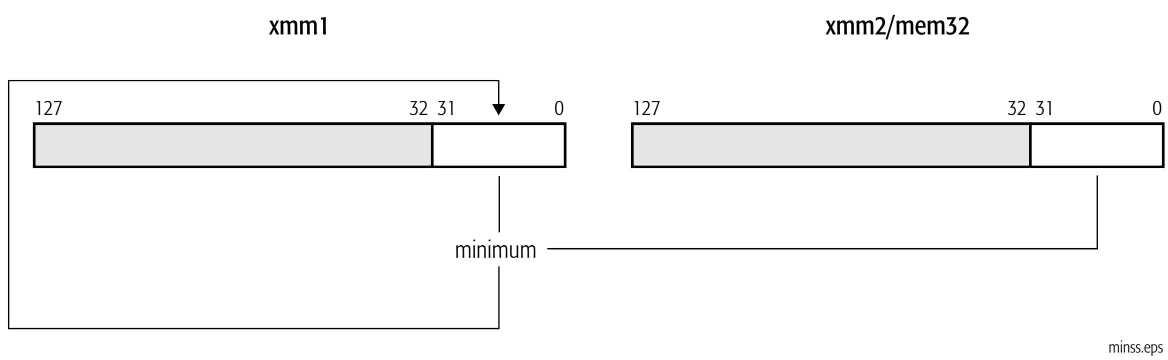
MINSS Minimum Scalar Single-Precision Floating-Point

Compares the single-precision floating-point value in the low-order 32 bits of the first source operand with the single-precision floating-point value in the low-order 32 bits of the second source operand and writes the numerically lesser of the two values in the low-order 32 bits of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or a 32-bit memory location. The three high-order doublewords of the destination XMM register are not modified.

If both source operands are equal to zero, the value in the second source operand is returned. If either operand is a NaN (SNaN or QNaN), and invalid-operation exceptions are masked, the second source operand is written to the destination.

The MINSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MINSS *xmm1*, *xmm2/mem32*	F3 0F 5D /*r*	Compares scalar single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the lesser of the two values in the destination XMM register.



Related Instructions

MAXPD, MAXPS, MAXSD, MAXSS, MINPD, MINPS, MINSD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN or QNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

MOVAPD Move Aligned Packed Double-Precision Floating-Point

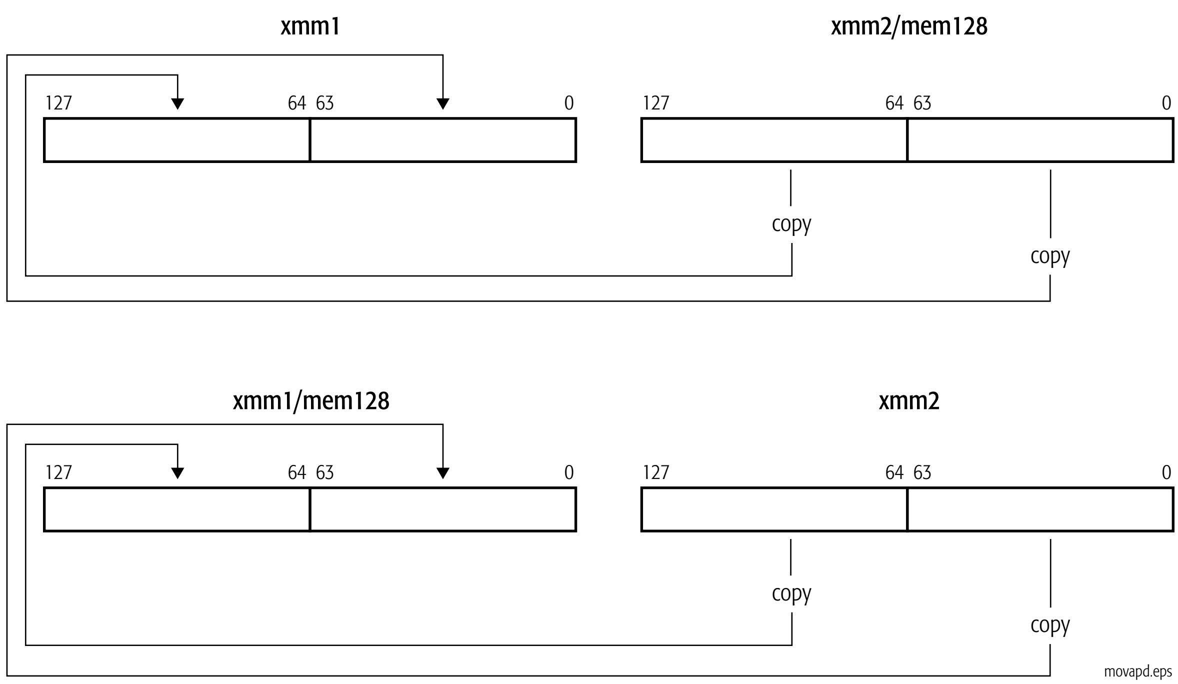
Moves two packed double-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

A memory operand that is not aligned on a 16-byte boundary causes a general-protection exception.

The MOVAPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MOVAPD *xmm1*, *xmm2/mem128*	66 0F 28 */r*	Moves packed double-precision floating-point value from an XMM register or 128-bit memory location to an XMM register.
MOVAPD *xmm1/mem128*, *xmm2*	66 0F 29 */r*	Moves packed double-precision floating-point value from an XMM register to an XMM register or 128-bit memory location.



Related Instructions

MOVHPD, MOVLPD, MOVMSKPD, MOVSD, MOVUPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVAPS — Move Aligned Packed Single-Precision Floating-Point

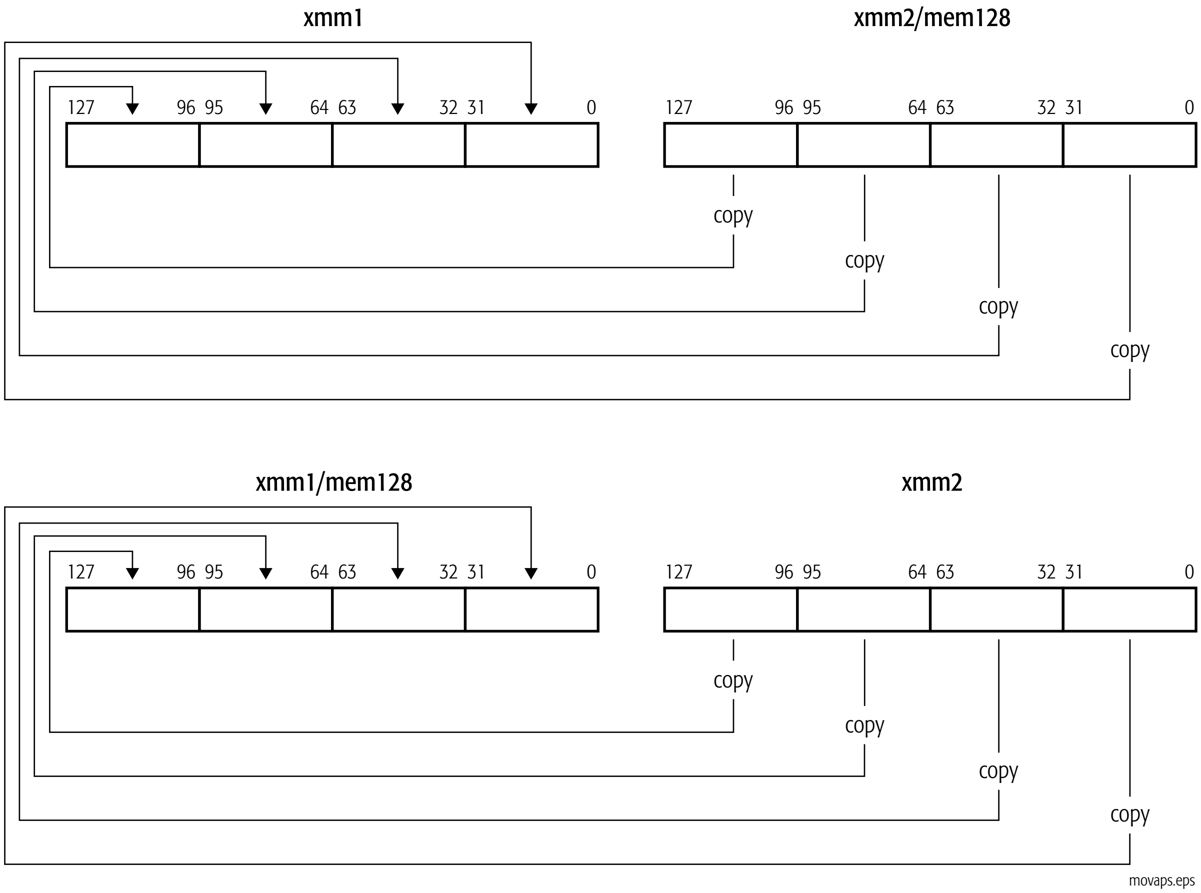
Moves four packed single-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

The MOVAPS instruction is an SSE instruction; check the status of EDX bit 25 returned by CPUID standard function 1 to verify that the processor supports this function. (See “CPUID” in Volume 3.)

A memory operand that is not aligned on a 16-byte boundary causes a general-protection exception.

Mnemonic	Opcode	Description
MOVAPS *xmm1*, *xmm2/mem128*	0F 28 */r*	Moves aligned packed single-precision floating-point value from an XMM register or 128-bit memory location to the destination XMM register.
MOVAPS *xmm1/mem128*, *xmm*2	0F 29 */r*	Moves aligned packed single-precision floating-point value from an XMM register to the destination XMM register or 128-bit memory location.



Related Instructions

MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

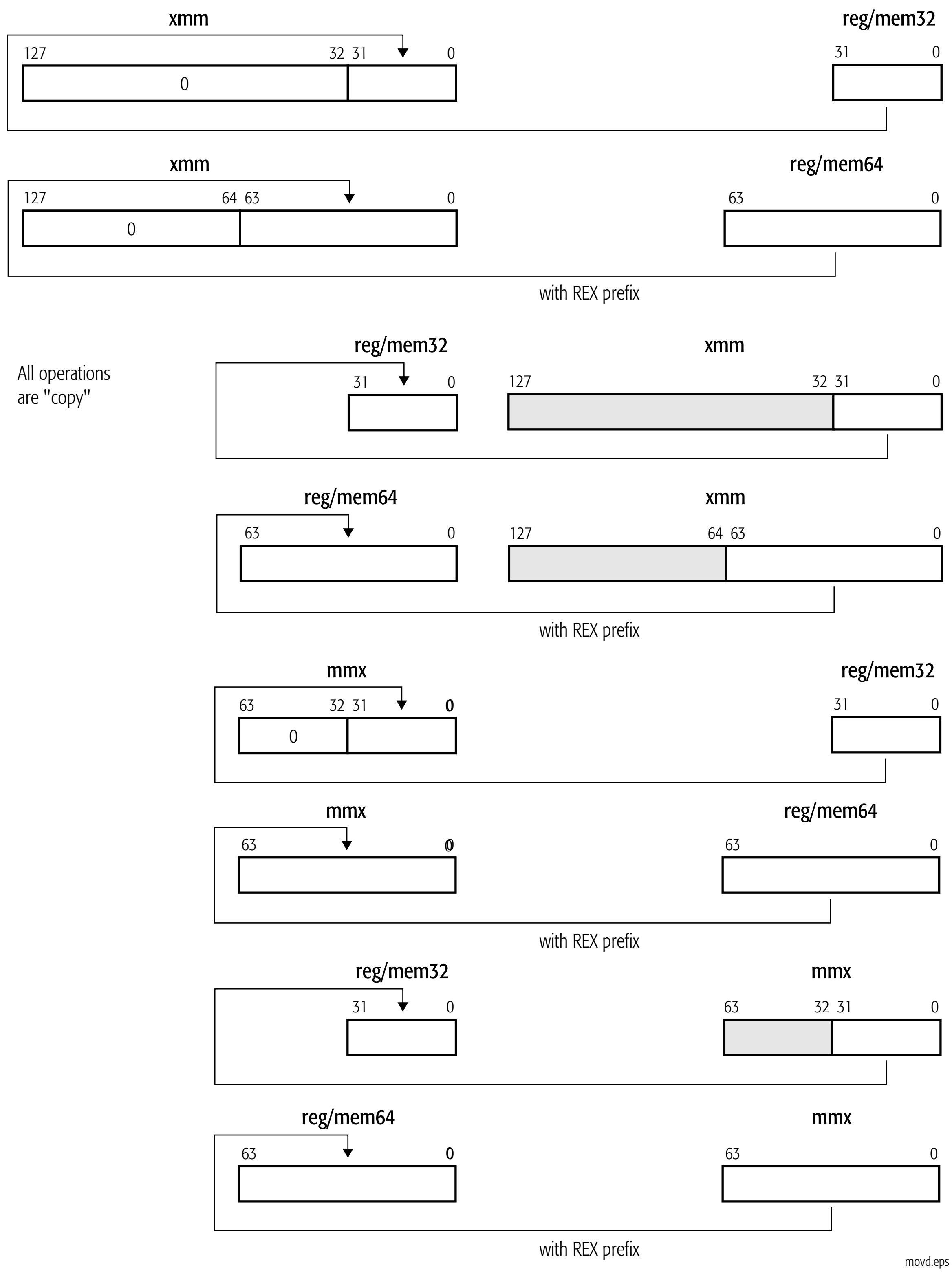
MOVD Move Doubleword or Quadword

Moves a 32-bit or 64-bit value in one of the following ways:

- from a 32-bit or 64-bit general-purpose register or memory location to the low-order 32 or 64 bits of an XMM register, with zero-extension to 128 bits
- from the low-order 32 or 64 bits of an XMM to a 32-bit or 64-bit general-purpose register or memory location
- from a 32-bit or 64-bit general-purpose register or memory location to the low-order 32 bits (with zero-extension to 64 bits) or the full 64 bits of an MMX register
- from the low-order 32 or the full 64 bits of an MMX register to a 32-bit or 64-bit general-purpose register or memory location

The MOVD instruction is an MMX instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MOVD *xmm*, *reg/mem32*	66 0F 6E /*r*	Move 32-bit value from a general-purpose register or 32-bit memory location to an XMM register.
MOVD *xmm*, *reg/mem64*	66 0F 6E /*r*	Move 64-bit value from a general-purpose register or 64-bit memory location to an XMM register.
MOVD *reg/mem32*, *xmm*	66 0F 7E /*r*	Move 32-bit value from an XMM register to a 32-bit general-purpose register or memory location.
MOVD *reg/mem64*, *xmm*	66 0F 7E /*r*	Move 64-bit value from an XMM register to a 64-bit general-purpose register or memory location.


xmm
reg/mem32
127
32 31
0
0
31
0
xmm
reg/mem64
127
64 63
0
0
63
0
with REX prefix
All operations
are "copy"
reg/mem32
xmm
31
0
127
32 31
0
reg/mem64
xmm
63
0
127
64 63
0
with REX prefix
mmx
reg/mem32
63
32 31
0
0
31
0
mmx
reg/mem64
63
0
63
0
with REX prefix
reg/mem32
mmx
31
0
63
32 31
0
reg/mem64
mmx
63
0
63
0
with REX prefix
movd.eps

Related Instructions

MOVDQA, MOVDQU, MOVDQ2Q, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions (All Modes)

Exception	Real	Virtual 8086	Protected	Description
Invalid opcode, #UD	X	X	X	The MMX™ instructions are not supported, as indicated by EDX bit 23 of CPUID standard function 1.
	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The instruction used XMM registers while CR4.OSFXSR=0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
x87 floating-point exception pending, #MF	X	X	X	An x87 floating-point exception was pending and the instruction referenced an MMX register.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

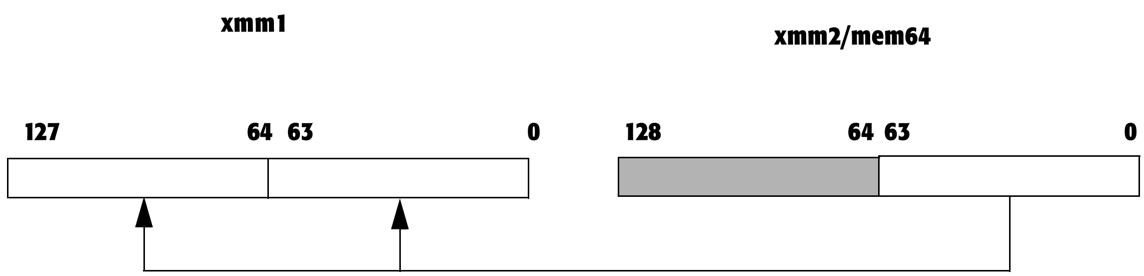
MOVDDUP Move Double-Precision and Duplicate

Moves a quadword value with its duplicate from the source operand to each quadword half of the XMM destination operand. The source operand may be an XMM register or the address of the least-significant byte of 64 bits of data in memory. When an XMM register is specified as the source operand, the lower 64-bits are duplicated and copied. When a memory address is specified, the 8 bytes of data at *mem64* are duplicated and loaded.

The MOVDDUP instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVDDUP *xmm1*, *xmm2/mem64*	F2 0F 12 /r	Moves two copies of the lower 64 bits of the source XMM or 128-bit memory operand to the lower and upper quadwords of the destination XMM register.

Related Instructions



Related Instructions

MOVSHDUP, MOVSLDUP

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

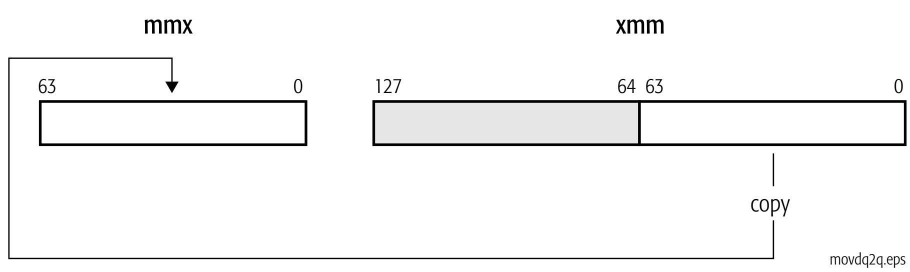
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVDQ2Q — Move Quadword to Quadword

Moves the low-order 64-bit value in an XMM register to a 64-bit MMX register.

The MOVDQ2Q instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVDQ2Q *mmx*, *xmm*	F2 0F D6 /r	Moves low-order 64-bit value from an XMM register to the destination MMX register.



Related Instructions

MOVD, MOVDQA, MOVDQU, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
General protection			X	The destination operand was in a non-writable segment.
x87 floating-point exception pending, #MF	X	X	X	An exception was pending due to an x87 floating-point instruction.

MOVDQA Move Aligned Double Quadword

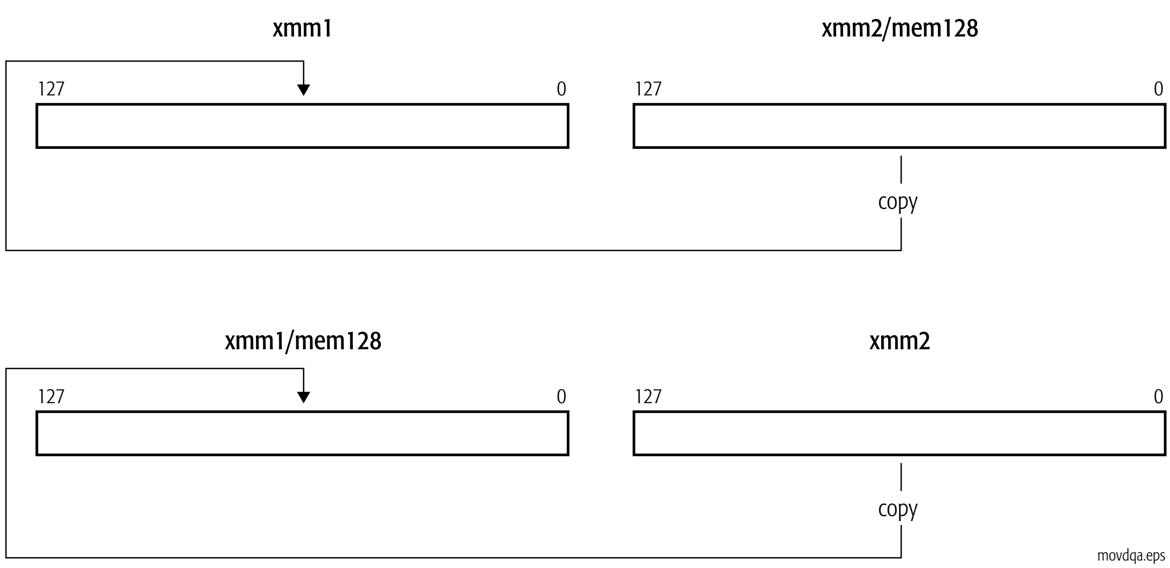
Moves an aligned 128-bit (double quadword) value:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to a 128-bit memory location or another XMM register.

A memory operand that is not aligned on a 16-byte boundary causes a general-protection exception.

The MOVDQA instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVDQA *xmm1*, *xmm2/mem128*	66 0F 6F */r*	Moves 128-bit value from an XMM register or 128-bit memory location to the destination XMM register.
MOVDQA *xmm1/mem128*, *xmm2*	66 0F 7F */r*	Moves 128-bit value from an XMM register to the destination XMM register or 128-bit memory location.



Related Instructions

MOVD, MOVDQU, MOVDQ2Q, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVDQU Move Unaligned Double Quadword

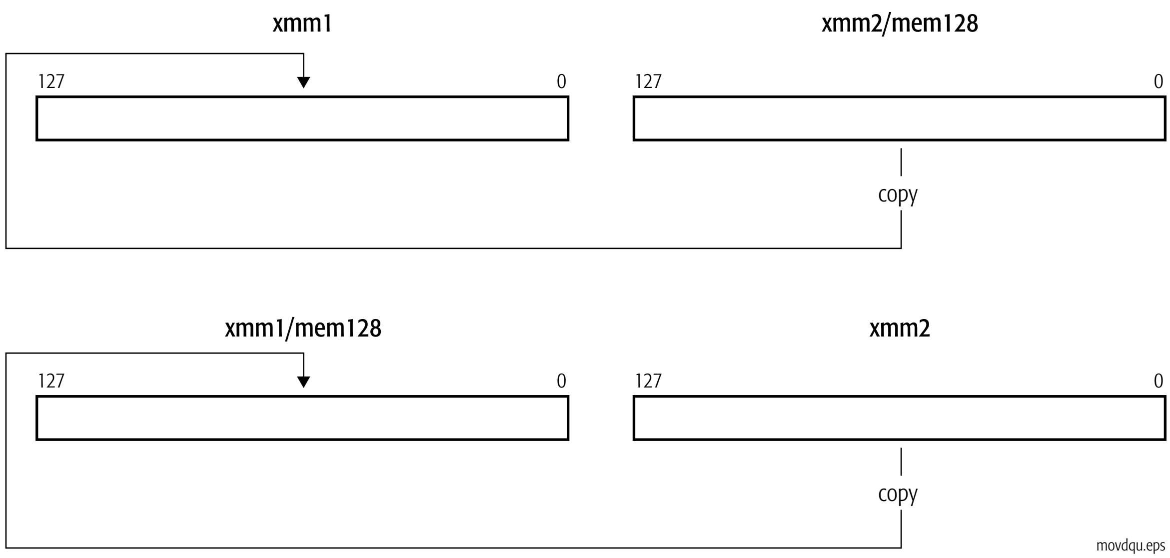
Moves an unaligned 128-bit (double quadword) value:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

The MOVDQU instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVDQU *xmm1*, *xmm2/mem128*	F3 0F 6F /*r*	Moves 128-bit value from an XMM register or unaligned 128-bit memory location to the destination XMM register.
MOVDQU *xmm1/mem128*, *xmm2*	F3 0F 7F /*r*	Moves 128-bit value from an XMM register to the destination XMM register or unaligned 128-bit memory location.



Related Instructions

MOVD, MOVDQA, MOVDQ2Q, MOVQ, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

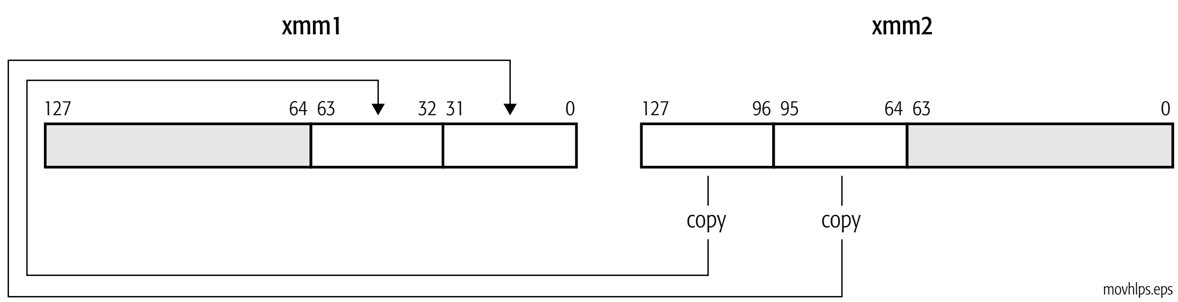
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indcated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned-memory reference was performed while alignment checking was enabled.

MOVHLPS — Move Packed Single-Precision Floating-Point High to Low

Moves two packed single-precision floating-point values from the high-order 64 bits of an XMM register to the low-order 64 bits of another XMM register. The high-order 64 bits of the destination XMM register are not modified.

The MOVHLPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVHLPS *xmm1*, *xmm2*	0F 12 /*r*	Moves two packed single-precision floating-point values from an XMM register to the destination XMM register.



Related Instructions

MOVAPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

MOVHPD Move High Packed Double-Precision Floating-Point

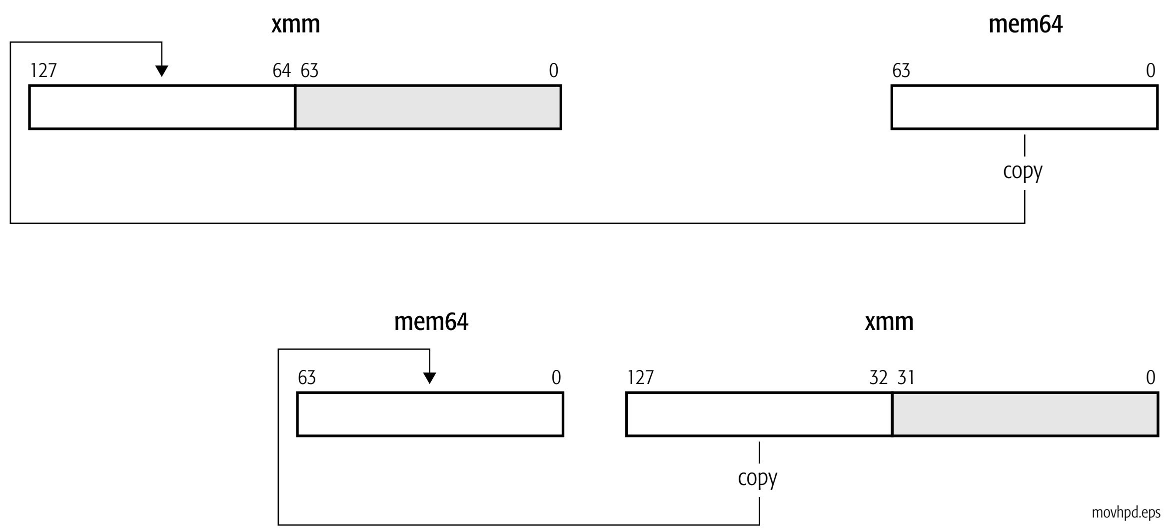
Moves a double-precision floating-point value:

- from a 64-bit memory location to the high-order 64 bits of an XMM register, or
- from the high-order 64 bits of an XMM register to a 64-bit memory location.

The low-order 64 bits of the destination XMM register are not modified.

The MOVHPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVHPD *xmm*, *mem64*	66 0F 16 /*r*	Moves double-precision floating-point value from a 64-bit memory location to an XMM register.
MOVHPD *mem64*, *xmm*	66 0F 17 /*r*	Moves double-precision floating-point value from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPD, MOVLPD, MOVMSKPD, MOVSD, MOVUPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVHPS Move High Packed Single-Precision Floating-Point

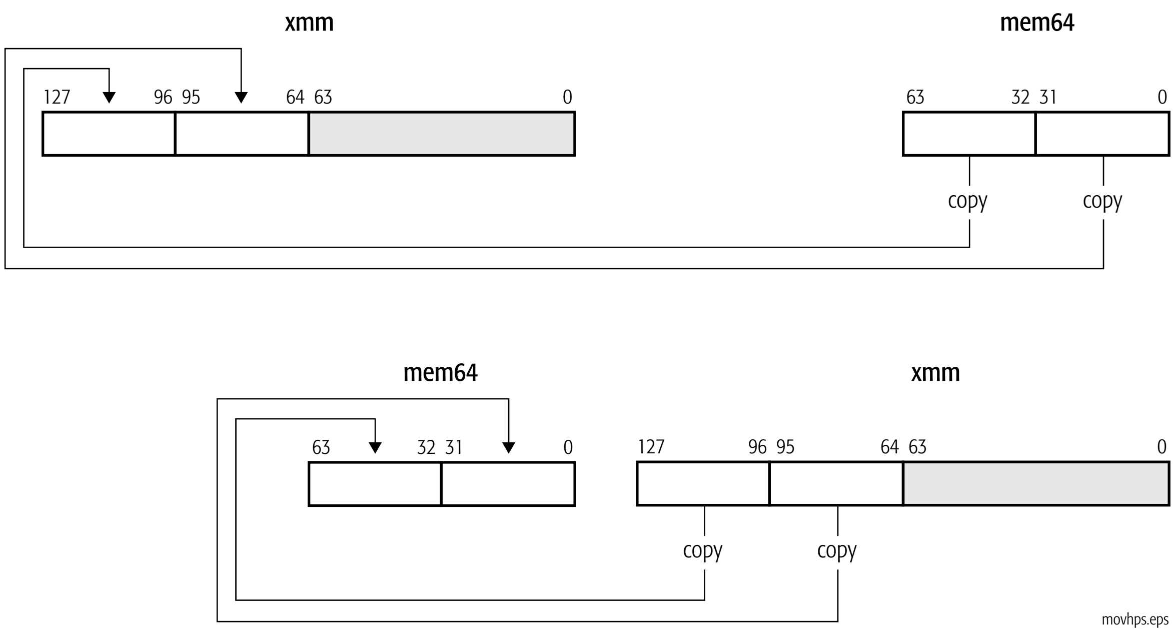
Moves two packed single-precision floating-point values:

- from a 64-bit memory location to the high-order 64 bits of an XMM register, or
- from the high-order 64 bits of an XMM register to a 64-bit memory location.

The low-order 64 bits of the destination XMM register are not modified.

The MOVHPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVHPS *xmm*, *mem64*	0F 16 /*r*	Moves two packed single-precision floating-point values from a 64-bit memory location to an XMM register.
MOVHPS *mem64*, *xmm*	0F 17 /*r*	Moves two packed single-precision floating-point values from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

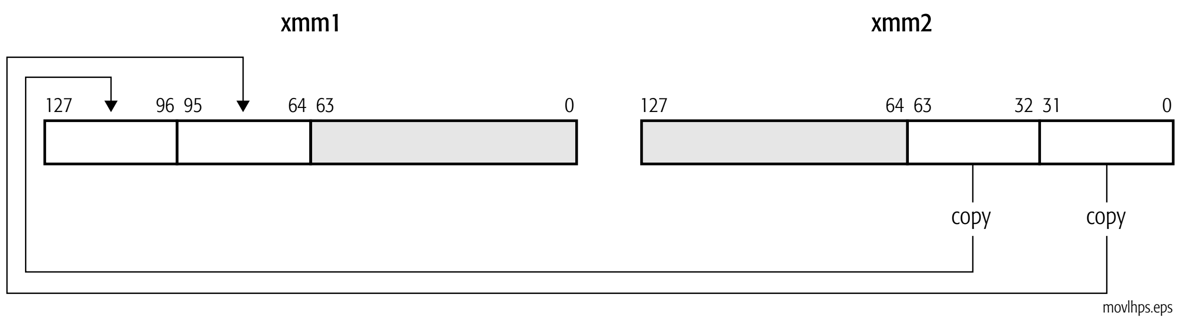
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVLHPS Move Packed Single-Precision Floating-Point Low to High

Moves two packed single-precision floating-point values from the low-order 64 bits of an XMM register to the high-order 64 bits of another XMM register. The low-order 64 bits of the destination XMM register are not modified.

The MOVLHPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MOVLHPS *xmm1*, *xmm2*	0F 16 */r*	Moves two packed single-precision floating-point values from an XMM register to another XMM register.



Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

MOVLPD — Move Low Packed Double-Precision Floating-Point

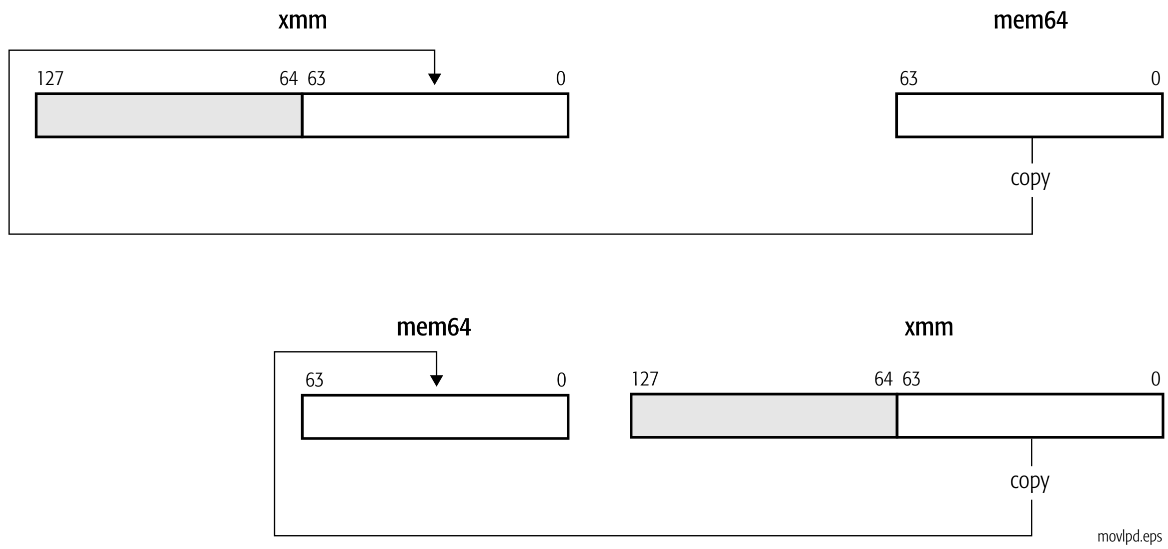
Moves a double-precision floating-point value:

- from a 64-bit memory location to the low-order 64 bits of an XMM register, or
- from the low-order 64 bits of an XMM register to a 64-bit memory location.

The high-order 64 bits of the destination XMM register are not modified.

The MOVLPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVLPD *xmm*, *mem64*	66 0F 12 /r	Moves double-precision floating-point value from a 64-bit memory location to an XMM register.
MOVLPD *mem64*, *xmm*	66 0F 13 /r	Moves double-precision floating-point value from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPD, MOVHPD, MOVMSKPD, MOVSD, MOVUPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVLPS — Move Low Packed Single-Precision Floating-Point

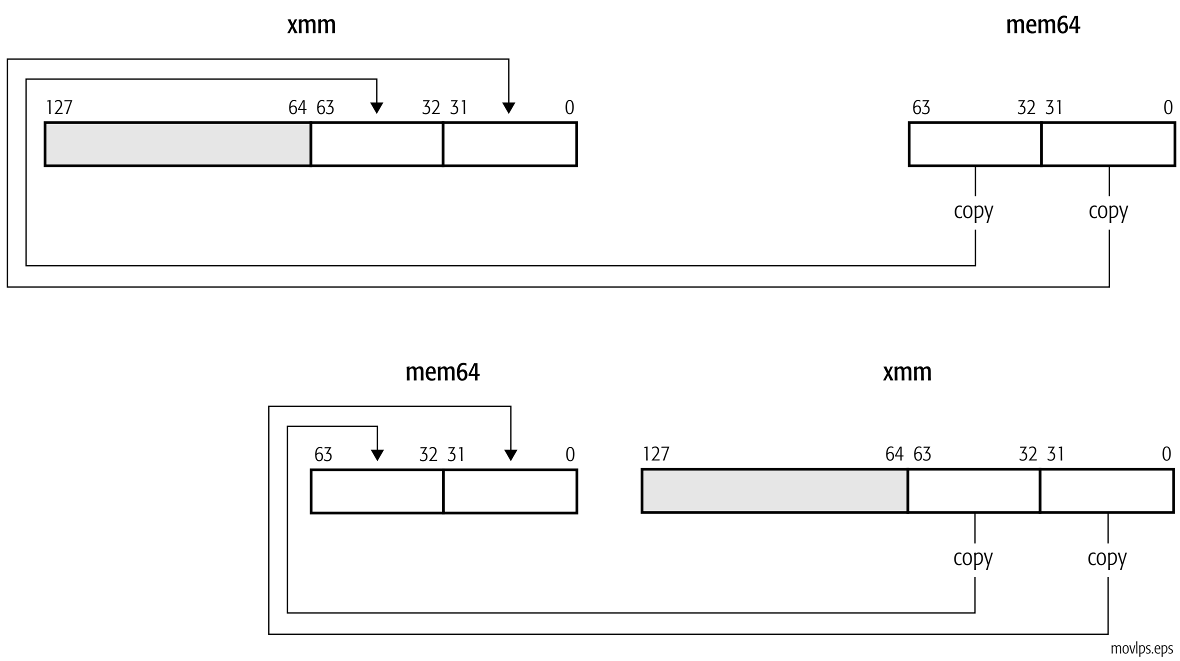
Moves two packed single-precision floating-point values:

- from a 64-bit memory location to the low-order 64 bits of an XMM register, or
- from the low-order 64 bits of an XMM register to a 64-bit memory location

The high-order 64 bits of the destination XMM register are not modified.

The MOVLPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVLPS *xmm*, *mem64*	0F 12 /*r*	Moves two packed single-precision floating-point values from a 64-bit memory location to an XMM register.
MOVLPS *mem64*, *xmm*	0F 13 /*r*	Moves two packed single-precision floating-point values from an XMM register to a 64-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVMSKPS, MOVSS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

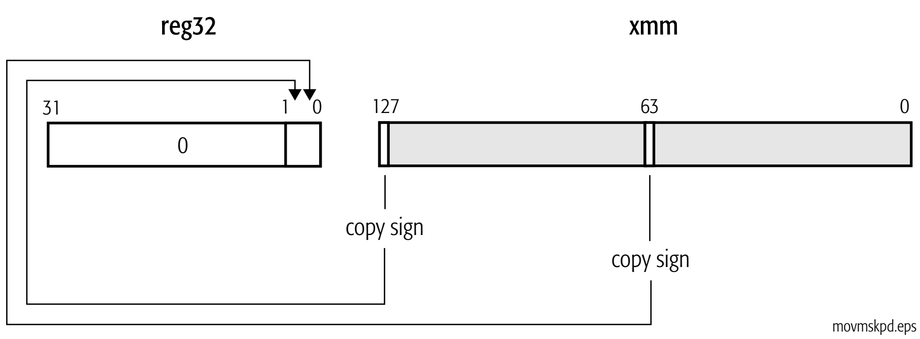
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of the control register (CR4) was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVMSKPD Extract Packed Double-Precision Floating-Point Sign Mask

Moves the sign bits of two packed double-precision floating-point values in an XMM register to the two low-order bits of a 32-bit general-purpose register, with zero-extension.

The MOVMSKPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVMSKPD *reg32*, *xmm*	66 0F 50 /*r*	Move sign bits in an XMM register to a 32-bit general-purpose register.



Related Instructions

MOVMSKPS, PMOVMSKB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

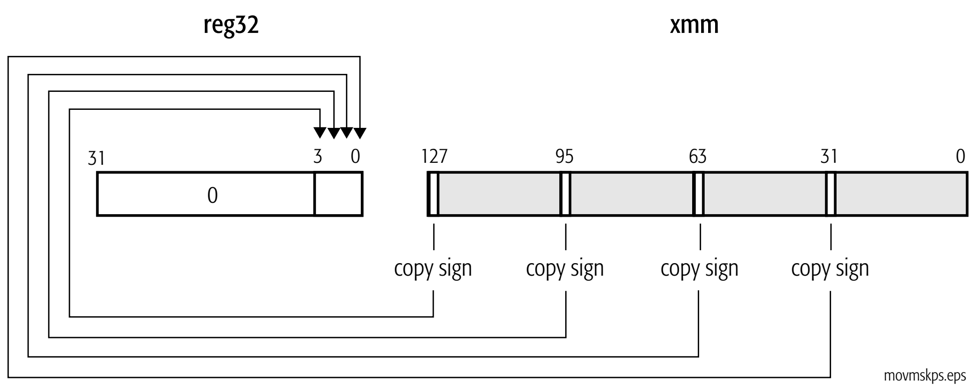
Exception (vector)	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

MOVMSKPS Extract Packed Single-Precision Floating-Point Sign Mask

Moves the sign bits of four packed single-precision floating-point values in an XMM register to the four low-order bits of a 32-bit general-purpose register, with zero-extension.

The MOVMSKPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVMSKPS *reg32*, *xmm*	0F 50 /*r*	Move sign bits in an XMM register to a 32-bit general-purpose register.



Related Instructions

MOVMSKPD, PMOVMSKB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

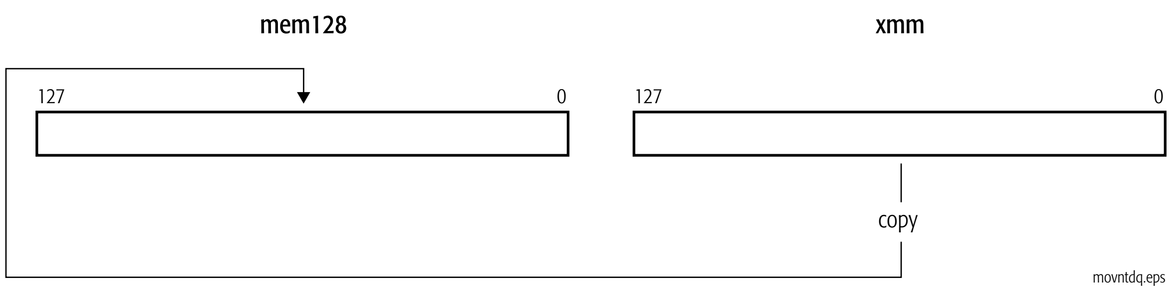
MOVNTDQ Move Non-Temporal Double Quadword

Stores a 128-bit (double quadword) XMM register value into a 128-bit memory location. This instruction indicates to the processor that the data is non-temporal, and is unlikely to be used again soon. The processor treats the store as a write-combining (WC) memory write, which minimizes cache pollution. The exact method by which cache pollution is minimized depends on the hardware implementation of the instruction. For further information, see "Memory Optimization" in Volume 1.

MOVNTDQ is weakly-ordered with respect to other instructions that operate on memory. Software should use an SFENCE instruction to force strong memory ordering of MOVNTDQ with respect to other stores.

The MOVNTDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MOVNTDQ *mem128*, *xmm*	66 0F E7 */r*	Stores a 128-bit XMM register value into a 128-bit memory location, minimizing cache pollution.



Related Instructions

MOVNTI, MOVNTPD, MOVNTPS, MOVNTQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

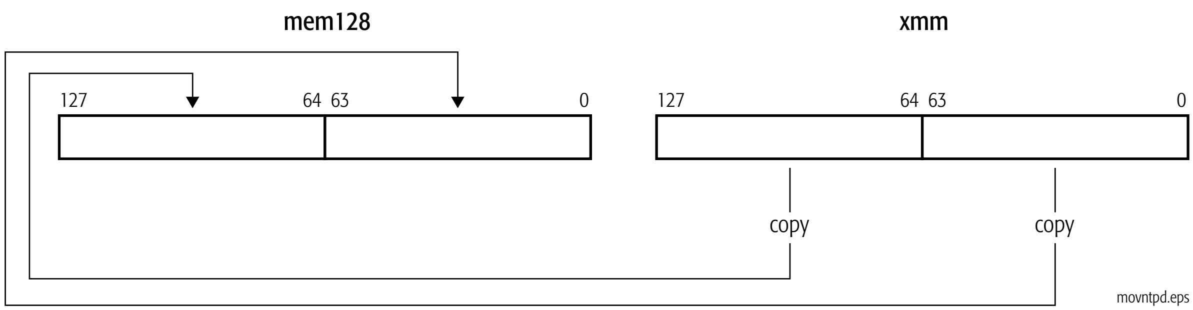
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (CR0.EM) was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (CR4.OSFXSR) was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (CR0.TS) was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from executing the instruction.

MOVNTPD Move Non-Temporal Packed Double-Precision Floating-Point

Stores two double-precision floating-point XMM register values into a 128-bit memory location. This instruction indicates to the processor that the data is non-temporal, and is unlikely to be used again soon. The processor treats the store as a write-combining (WC) memory write, which minimizes cache pollution. The exact method by which cache pollution is minimized depends on the hardware implementation of the instruction. For further information, see “Memory Optimization” in Volume 1.

The MOVNTPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVNTPD *mem128*, *xmm*	66 0F 2B /*r*	Stores two packed double-precision floating-point XMM register values into a 128-bit memory location, minimizing cache pollution.



MOVNTPD is weakly-ordered with respect to other instructions that operate on memory. Software should use an SFENCE instruction to force strong memory ordering of MOVNTPD with respect to other stores.

Related Instructions

MOVNTDQ, MOVNTI, MOVNTPS, MOVNTQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

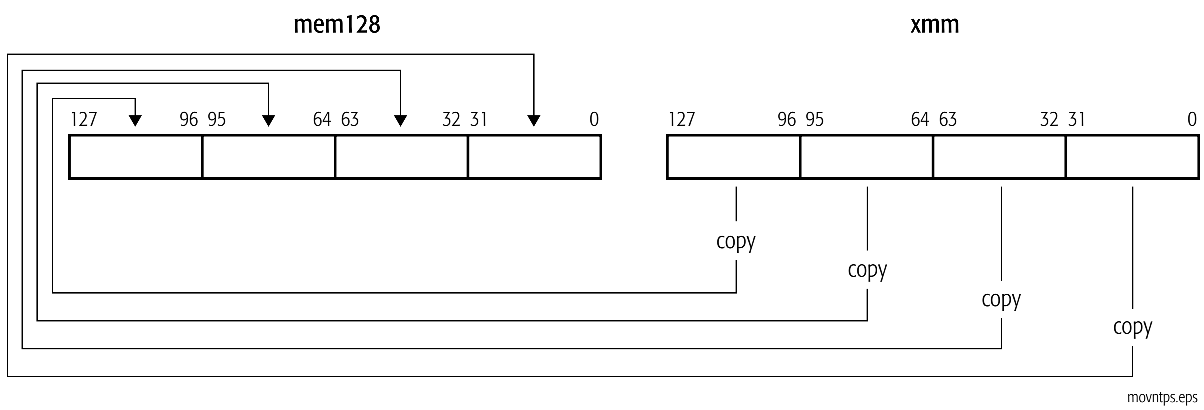
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (CR0.EM) was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (CR4.OSFXSR) was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (CR0.TS) was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from executing the instruction.

MOVNTPS Move Non-Temporal Packed Single-Precision Floating-Point

Stores four single-precision floating-point XMM register values into a 128-bit memory location. This instruction indicates to the processor that the data is non-temporal, and is unlikely to be used again soon. The processor treats the store as a write-combining (WC) memory write, which minimizes cache pollution. The exact method by which cache pollution is minimized depends on the hardware implementation of the instruction. For further information, see “Memory Optimization” in Volume 1.

The MOVNTPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVNTPS *mem128*, *xmm*	0F 2B /*r*	Stores four packed single-precision floating-point XMM register values into a 128-bit memory location, minimizing cache pollution.



MOVNTPD is weakly-ordered with respect to other instructions that operate on memory. Software should use an SFENCE instruction to force strong memory ordering of MOVNTPD with respect to other stores.

Related Instructions

MOVNTDQ, MOVNTI, MOVNTPD, MOVNTQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (CR0.EM) was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (CR4.OSFXSR) was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (CR0.TS) was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from executing the instruction.

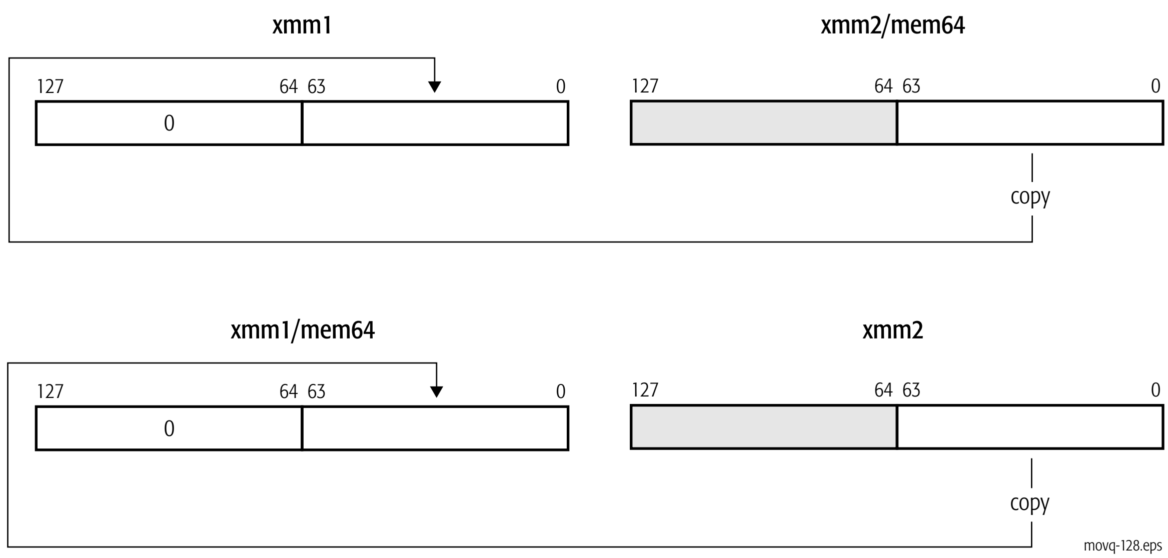
MOVQ Move Quadword

Moves a 64-bit value in one of the following ways:

- from the low-order 64 bits of an XMM register or a 64-bit memory location to the low-order 64 bits of another XMM register, with zero-extension to 128 bits
- from the low-order 64 bits of an XMM register to the low-order 64 bits of another XMM register, with zero-extension to 128 bits or to a 64-bit memory location

The MOVQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVQ *xmm1*, *xmm2/mem64*	F3 0F 7E /r	Moves 64-bit value from an XMM register or memory location to an XMM register.
MOVQ *xmm1/mem64*, *xmm2*	66 0F D6 /r	Moves 64-bit value from an XMM register to an XMM register or memory location.



Related Instructions

MOVD, MOVDQA, MOVDQU, MOVDQ2Q, MOVQ2DQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

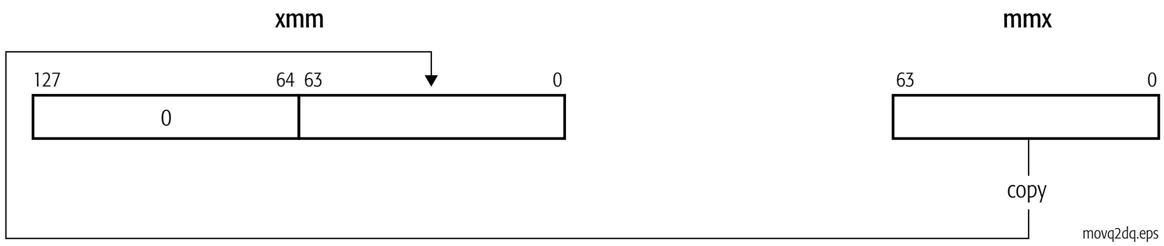
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVQ2DQ Move Quadword to Quadword

Moves a 64-bit value from an MMX register to the low-order 64 bits of an XMM register, with zero-extension to 128 bits.

The MOVQ2DQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVQ2DQ *xmm*, *mmx*	F3 0F D6 /r	Moves 64-bit value from an MMX™ register to an XMM register.



Related Instructions

MOVD, MOVDQA, MOVDQU, MOVDQ2Q, MOVQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
x87 floating-point exception pending, #MF	X	X	X	An exception was pending due to an x87 floating-point instruction.

MOVSD Move Scalar Double-Precision Floating-Point

Moves a scalar double-precision floating-point value:

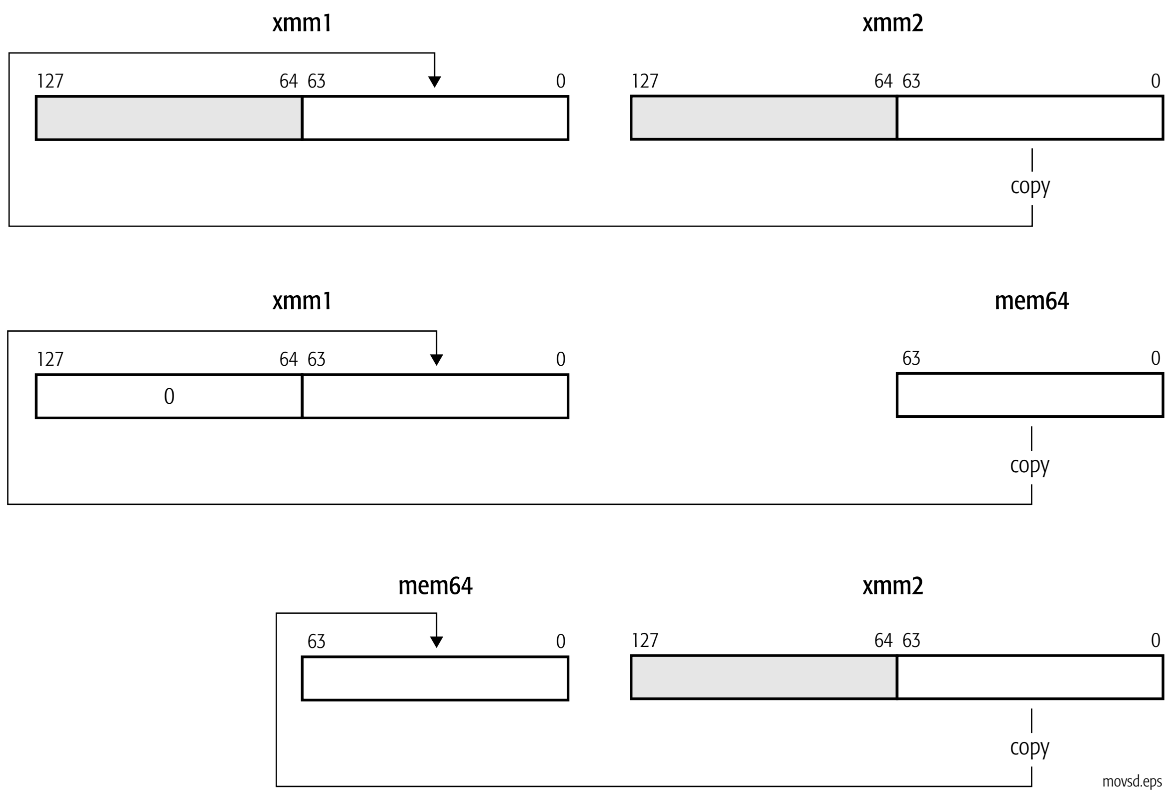
- from the low-order 64 bits of an XMM register or a 64-bit memory location to the low-order 64 bits of another XMM register, or
- from the low-order 64 bits of an XMM register to the low-order 64 bits of another XMM register or a 64-bit memory location.

If the source operand is an XMM register, the high-order 64 bits of the destination XMM register are not modified. If the source operand is a memory location, the high-order 64 bits of the destination XMM register are cleared to all 0s.

This MOVSD instruction should not be confused with the MOVSD (move string doubleword) instruction with the same mnemonic in the general-purpose instruction set. Assemblers can distinguish the instructions by the number and type of operands.

The MOVSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVSD *xmm1*, *xmm2/mem64*	F2 0F 10 /*r*	Moves double-precision floating-point value from an XMM register or 64-bit memory location to an XMM register.
MOVSD *xmm1/mem64*, *xmm2*	F2 0F 11 /*r*	Moves double-precision floating-point value from an XMM register to an XMM register or 64-bit memory location.



Related Instructions

MOVAPD, MOVHPD, MOVLPD, MOVMSKPD, MOVUPD

rFLAGS Affected

None.

MXCSR Flags Affected

None.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.

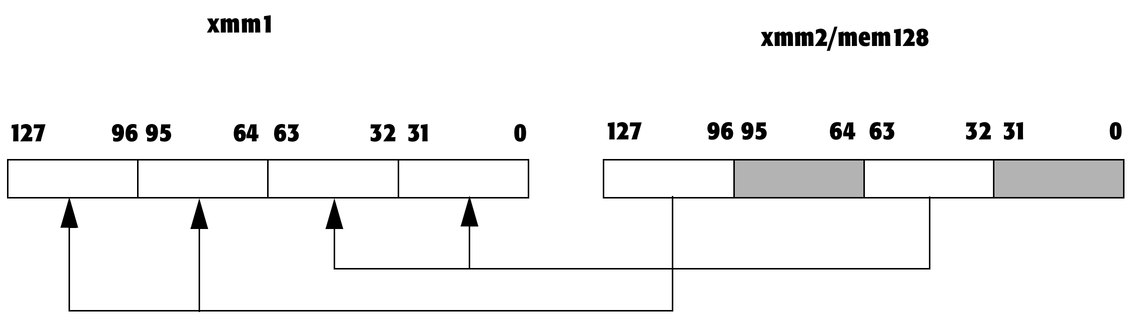
Exception	Real	Virtual 8086	Protected	Cause of Exception
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVSHDUP Move Single-Precision High and Duplicate

Moves two copies of the second doubleword of data in the source XMM register or 128-bit memory operand to bits 31–0 and bits 63–32 of the destination XMM register; moves two copies of the fourth doubleword of data in the source operand to bits 95–64 and bits 127–96 of the destination XMM register.

The MOVSHDUP instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVSHDUP *xmm1*, *xmm2/mem128*	F3 0F 16 /r	Copies the second 32-bits from the source operand to the first and second 32-bit segments of the destination XMM register; copies the fourth 32-bits from the source operand to the third and fourth 32-bit segments of the destination XMM register.



Related Instructions

MOVDDUP, MOVSLDUP

rFLAGS Affected

None.

MXCSR Flags Affected

None.

Exceptions

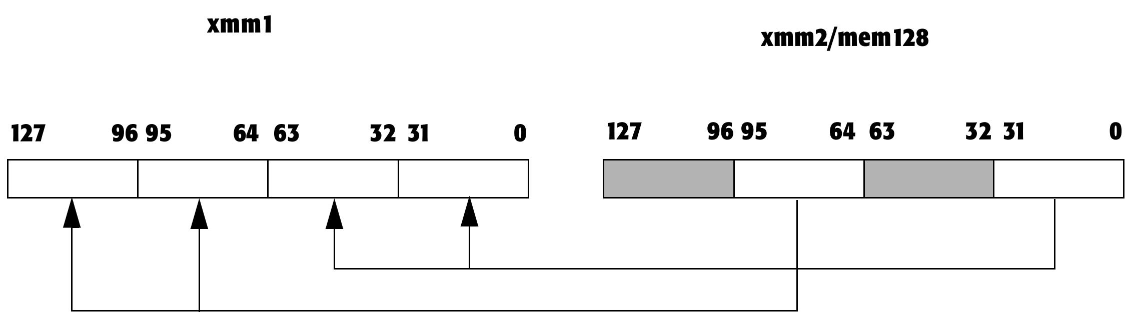
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVSLDUP Move Single-Precision Low and Duplicate

Moves two copies of the first doubleword of data in the source XMM register or 128-bit memory operand to bits 31–0 and bits 32–63 of the destination XMM register and moves two copies of the third doubleword of data in the source operand to bits 95–64 and bits 127–96 of the destination XMM register.

The MOVSLDUP instruction is an SSE3 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVSLDUP *xmm1*, *xmm2/mem128*	F3 0F 12 /r	Copies the first 32-bits from the source operand to the first and second 32-bit segments of the destination XMM register; copies the third 32-bits from the source operand to the third and fourth 32-bit segments of the destination XMM register.



Related Instructions

MOVDDUP, MOVSHDUP

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE3 instructions are not supported, as indicated by ECX bit 0 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

MOVSS Move Scalar Single-Precision Floating-Point

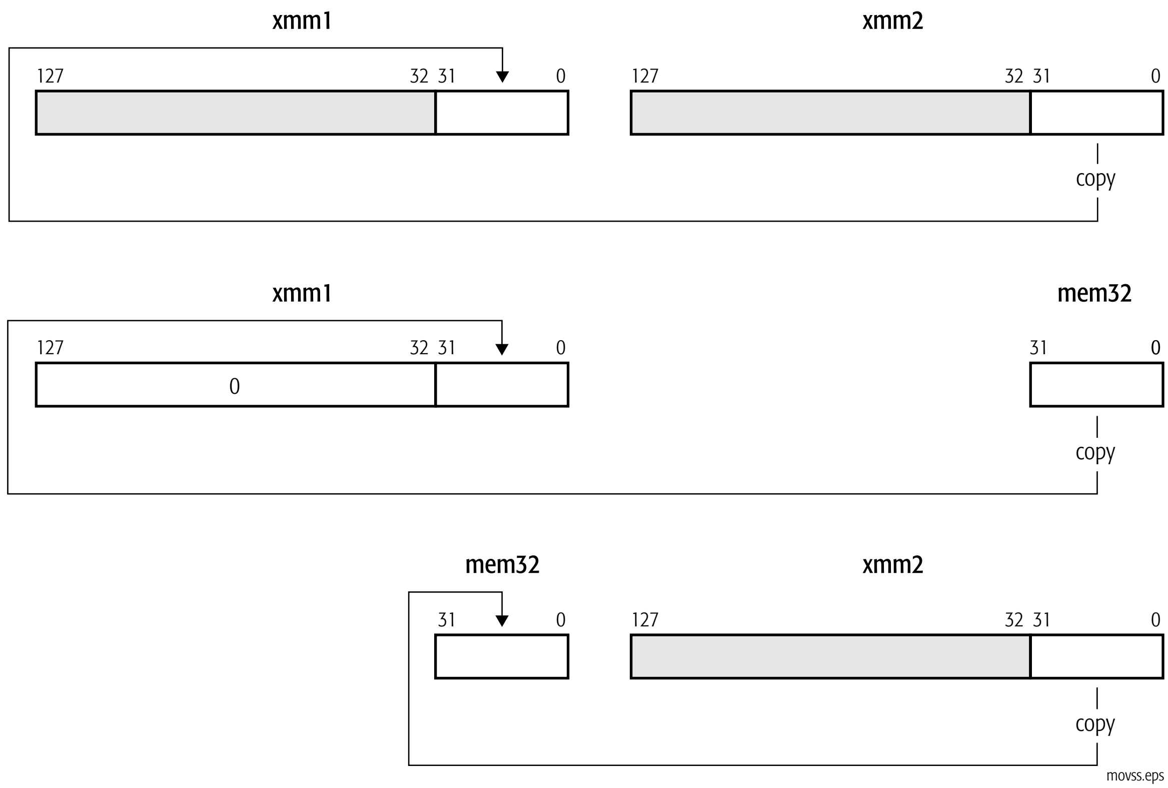
Moves a scalar single-precision floating-point value:

- from the low-order 32 bits of an XMM register or a 32-bit memory location to the low-order 32 bits of another XMM register, or
- from a 32-bit memory location to the low-order 32 bits of an XMM register, with zero-extension to 128 bits.

If the source operand is an XMM register, the high-order 96 bits of the destination XMM register are not modified. If the source operand is a memory location, the high-order 96 bits of the destination XMM register are cleared to all 0s.

The MOVSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVSS *xmm1*, *xmm2/mem32*	F3 0F 10 /*r*	Moves single-precision floating-point value from an XMM register or 32-bit memory location to an XMM register.
MOVSS *xmm1/mem32*, *xmm2*	F3 0F 11 /*r*	Moves single-precision floating-point value from an XMM register to an XMM register or 32-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVUPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

MOVUPD Move Unaligned Packed Double-Precision Floating-Point

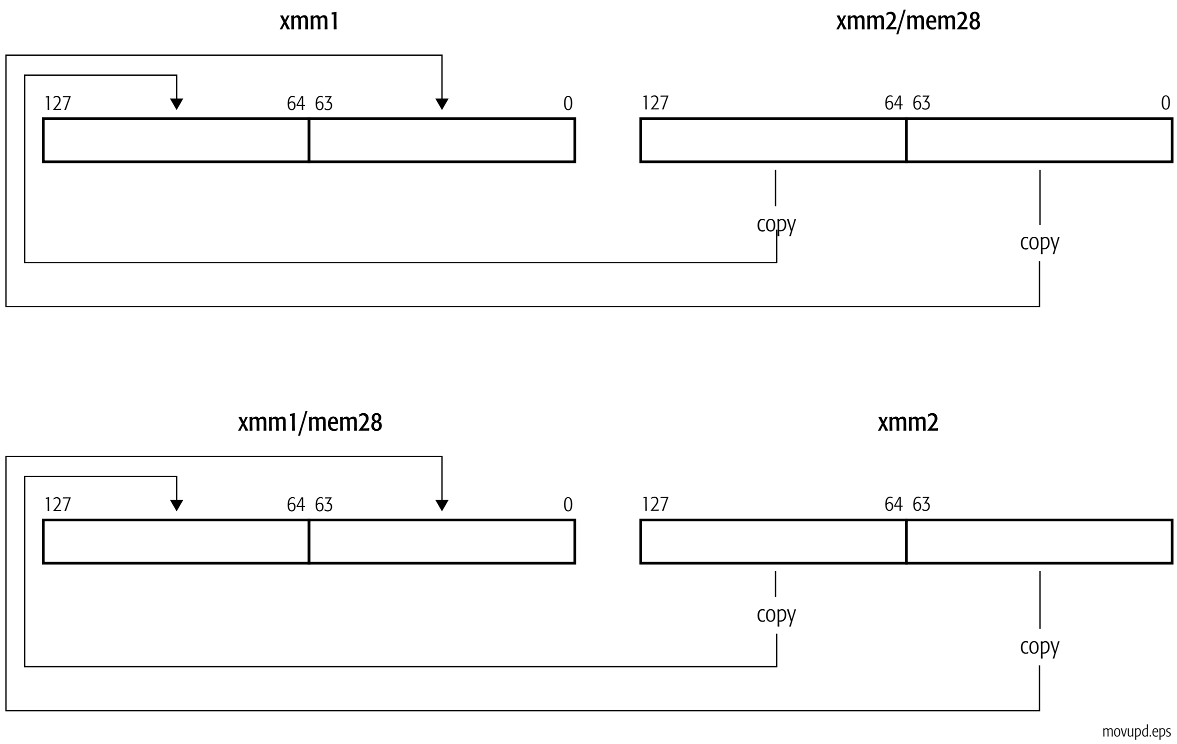
Moves two packed double-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

The MOVUPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVUPD *xmm1*, *xmm2/mem128*	66 0F 10 /*r*	Moves two packed double-precision floating-point values from an XMM register or unaligned 128-bit memory location to an XMM register.
MOVUPD *xmm1/mem128*, *xmm2*	66 0F 11 /*r*	Moves two packed double-precision floating-point values from an XMM register to an XMM register or unaligned 128-bit memory location.



Related Instructions

MOVAPD, MOVHPD, MOVLPD, MOVMSKPD, MOVSD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned-memory reference was performed while alignment checking was enabled.

MOVUPS Move Unaligned Packed Single-Precision Floating-Point

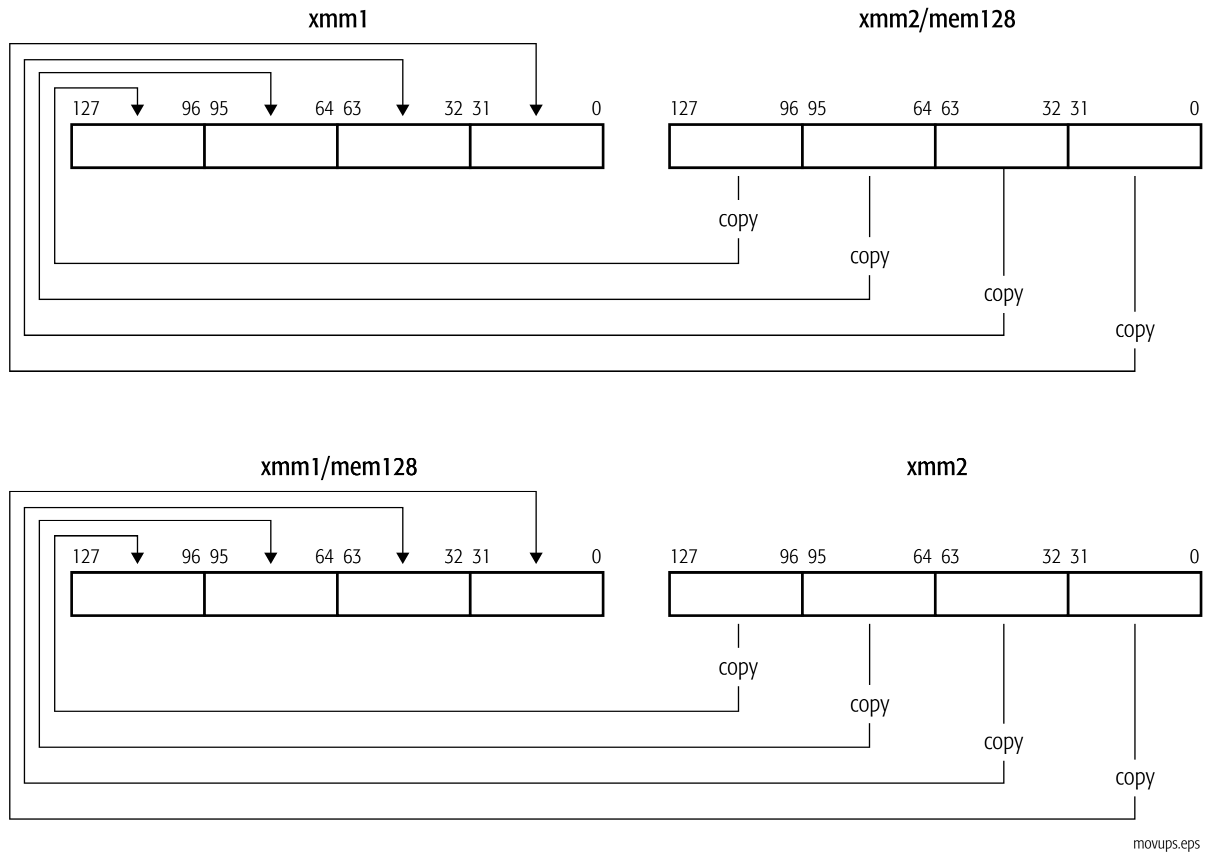
Moves four packed single-precision floating-point values:

- from an XMM register or 128-bit memory location to another XMM register, or
- from an XMM register to another XMM register or 128-bit memory location.

Memory operands that are not aligned on a 16-byte boundary do not cause a general-protection exception.

The MOVUPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MOVUPS *xmm1*, *xmm2/mem128*	0F 10 */r*	Moves four packed single-precision floating-point values from an XMM register or unaligned 128-bit memory location to an XMM register.
MOVUPS *xmm1/mem128*, *xmm2*	0F 11 */r*	Moves four packed single-precision floating-point values from an XMM register to an XMM register or unaligned 128-bit memory location.



Related Instructions

MOVAPS, MOVHLPS, MOVHPS, MOVLHPS, MOVLPS, MOVMSKPS, MOVSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

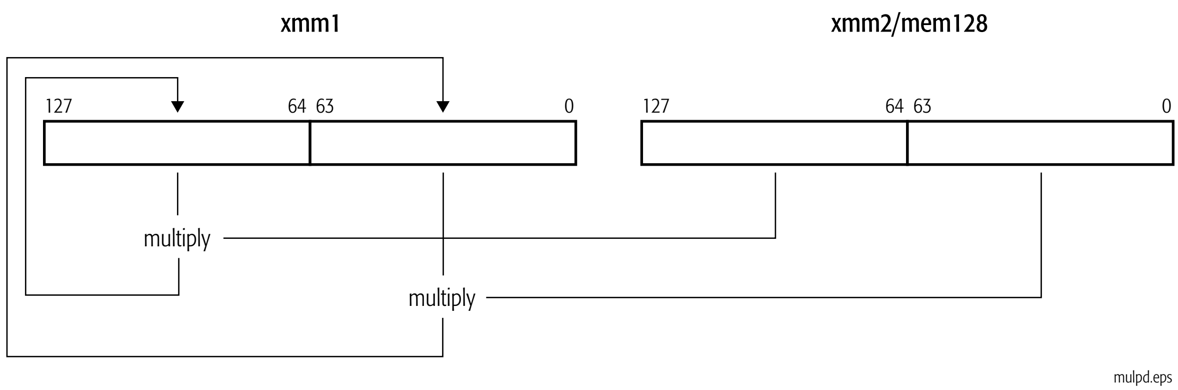
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned-memory reference was performed while alignment checking was enabled.

MULPD Multiply Packed Double-Precision Floating-Point

Multiplies each of the two packed double-precision floating-point values in the first source operand by the corresponding packed double-precision floating-point value in the second source operand and writes the result of each multiplication operation in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The MULPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MULPD *xmm1*, *xmm2/mem128*	66 0F 59 */r*	Multiplies packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the results in the destination XMM register.



Related Instructions

MULPS, MULSD, MULSS, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was multiplied by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.

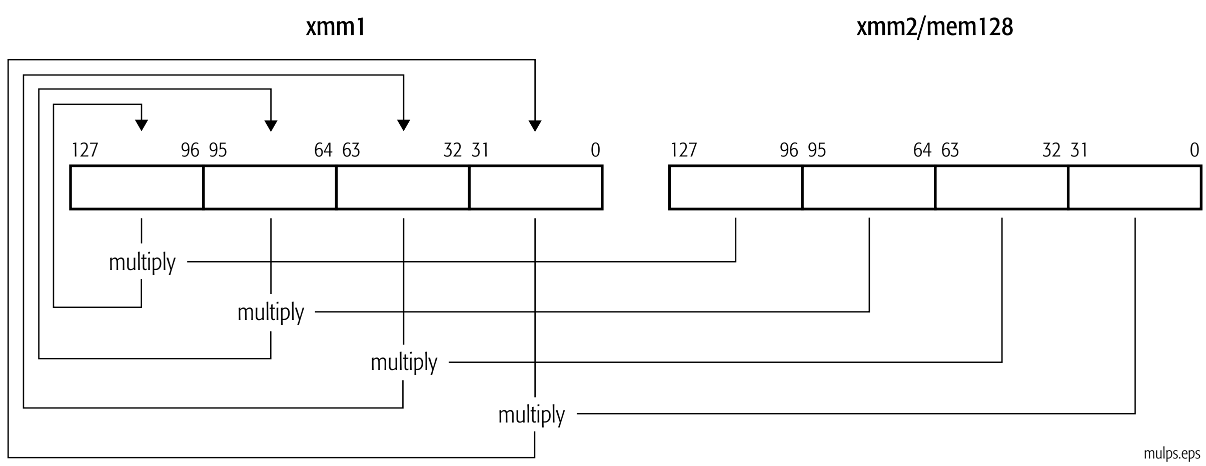
Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

MULPS Multiply Packed Single-Precision Floating-Point

Multiplies each of the four packed single-precision floating-point values in first source operand by the corresponding packed single-precision floating-point value in the second source operand and writes the result of each multiplication operation in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The MULPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
MULPS *xmm1*, *xmm2/mem128*	0F 59 */r*	Multiplies packed single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and writes the results in the destination XMM register.



Related Instructions

MULPD, MULSD, MULSS, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was multipled by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.

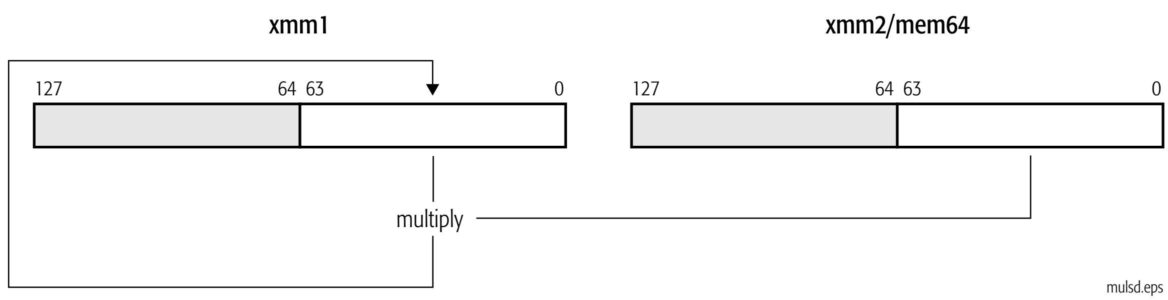
Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

MULSD Multiply Scalar Double-Precision Floating-Point

Multiplies the double-precision floating-point value in the low-order quadword of first source operand by the double-precision floating-point value in the low-order quadword of the second source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location.

The MULSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MULSD *xmm1*, *xmm2/mem64*	F2 0F 59 /*r*	Multiplies low-order double-precision floating-point values in an XMM register and another XMM register or 64-bit memory location and writes the result in the low-order quadword of the destination XMM register.



Related Instructions

MULPD, MULPS, MULSS, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was multipled by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.

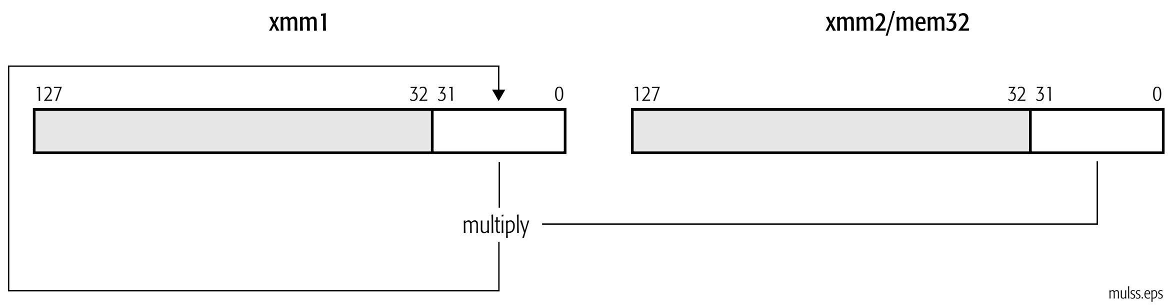
Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

MULSS — Multiply Scalar Single-Precision Floating-Point

Multiplies the single-precision floating-point value in the low-order doubleword of first source operand by the single-precision floating-point value in the low-order doubleword of the second source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location.

The MULSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
MULSS *xmm1*, *xmm2/mem32*	F3 0F 59 */r*	Multiplies low-order single-precision floating-point values in an XMM register and another XMM register or 32-bit memory location and writes the result in the low-order doubleword of the destination XMM register.



Related Instructions

MULPD, MULPS, MULSD, PFMUL

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	±Zero was multipled by ±infinity.
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.

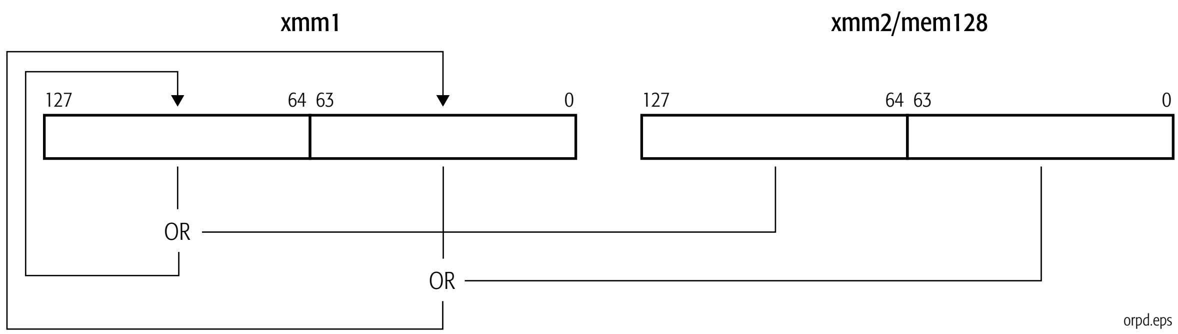
Exception	Real	Virtual 8086	Protected	Cause of Exception
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

ORPD — Logical Bitwise OR Packed Double-Precision Floating-Point

Performs a bitwise logical OR of the two packed double-precision floating-point values in the first source operand and the corresponding two packed double-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ORPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ORPD *xmm1*, *xmm2/mem128*	66 0F 56 */r*	Performs bitwise logical OR of two packed double-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPS, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

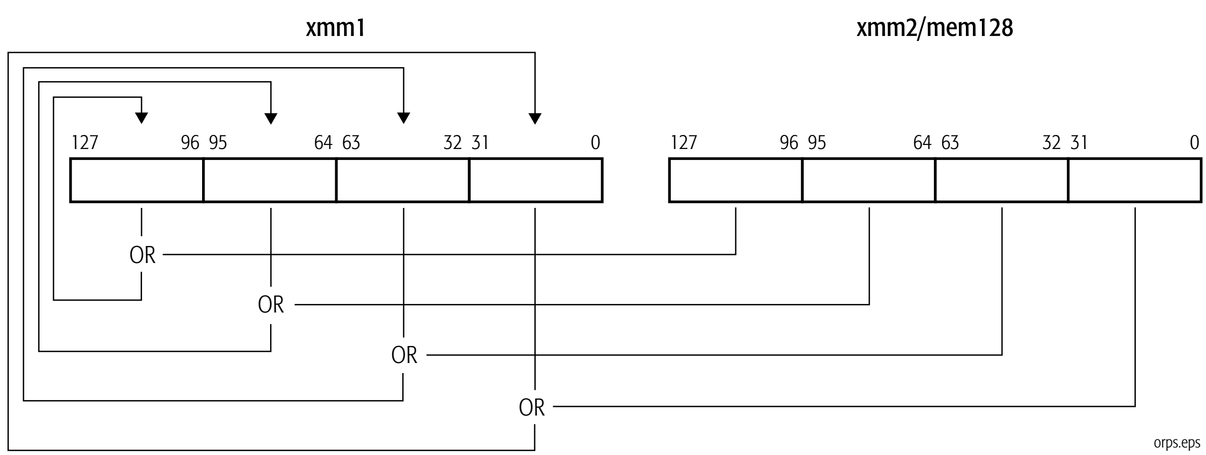
ORPS

Logical Bitwise OR Packed Single-Precision Floating-Point

Performs a bitwise logical OR of the four packed single-precision floating-point values in the first source operand and the corresponding four packed single-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The ORPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
ORPS *xmm1*, *xmm2/mem128*	0F 56 /*r*	Performs bitwise logical OR of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPD, XORPD, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

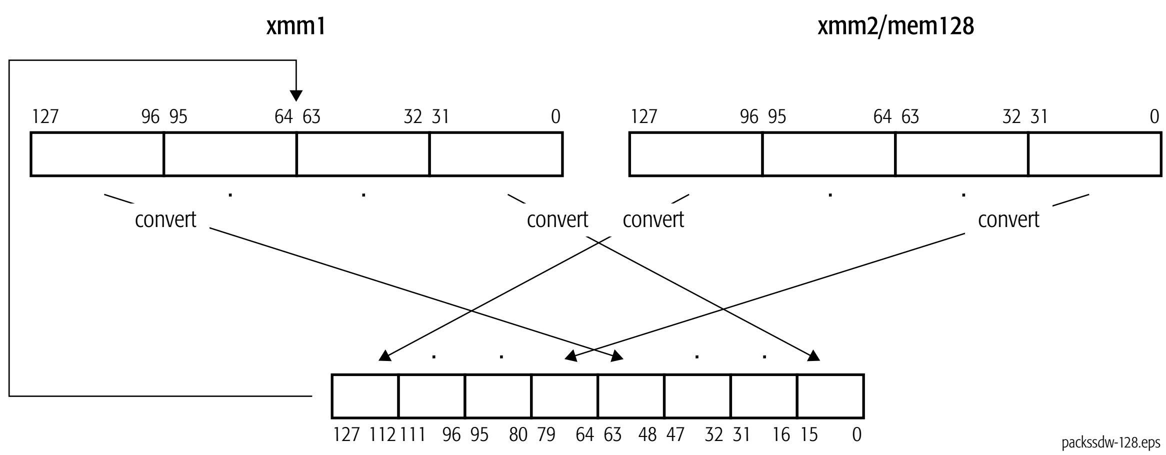
PACKSSDW Pack with Saturation Signed Doubleword to Word

Converts each 32-bit signed integer in the first and second source operands to a 16-bit signed integer and packs the converted values into words in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Converted values from the first source operand are packed into the low-order words of the destination, and the converted values from the second source operand are packed into the high-order words of the destination.

The PACKSSDW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PACKSSDW *xmm1*, *xmm2/mem128*	66 0F 6B /*r*	Packs 32-bit signed integers in an XMM register and another XMM register or 128-bit memory location into 16-bit signed integers in an XMM register.



For each packed value in the destination, if the value is larger than the largest signed 16-bit integer, it is saturated to 7FFFh, and if the value is smaller than the smallest signed 16-bit integer, it is saturated to 8000h.

Related Instructions

PACKSSWB, PACKUSWB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

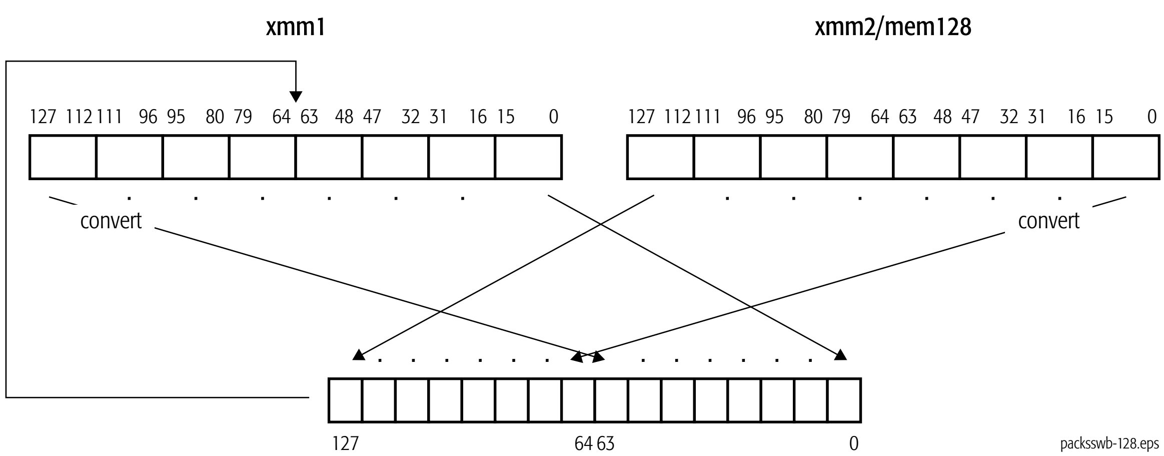
PACKSSWB Pack with Saturation Signed Word to Byte

Converts each 16-bit signed integer in the first and second source operands to an 8-bit signed integer and packs the converted values into bytes in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Converted values from the first source operand are packed into the low-order bytes of the destination, and the converted values from the second source operand are packed into the high-order bytes of the destination.

The PACKSSWB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PACKSSWB *xmm1*, *xmm2/mem128*	66 0F 63 */r*	Packs 16-bit signed integers in an XMM register and another XMM register or 128-bit memory location into 8-bit signed integers in an XMM register.



For each packed value in the destination, if the value is larger than the largest signed 8-bit integer, it is saturated to 7Fh, and if the value is smaller than the smallest signed 8-bit integer, it is saturated to 80h.

Related Instructions

PACKSSDW, PACKUSWB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

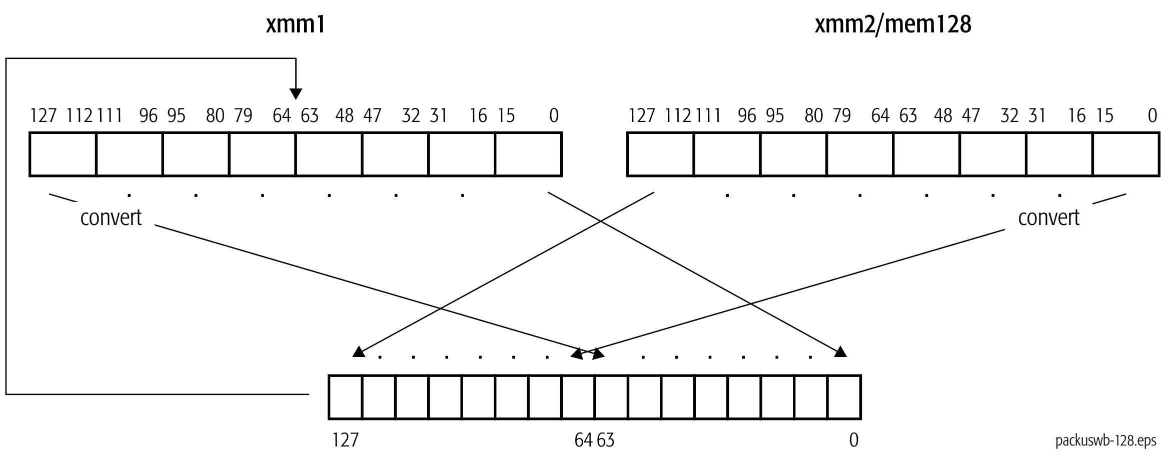
PACKUSWB Pack with Saturation Signed Word to Unsigned Byte

Converts each 16-bit signed integer in the first and second source operands to an 8-bit unsigned integer and packs the converted values into bytes in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

Converted values from the first source operand are packed into the low-order bytes of the destination, and the converted values from the second source operand are packed into the high-order bytes of the destination.

The PACKUSWB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PACKUSWB *xmm1*, *xmm2/mem128*	66 0F 67 */r*	Packs 16-bit signed integers in an XMM register and another XMM register or 128-bit memory location into 8-bit unsigned integers in an XMM register.



For each packed value in the destination, if the value is larger than the largest unsigned 8-bit integer, it is saturated to FFh, and if the value is smaller than the smallest unsigned 8-bit integer, it is saturated to 00h.

Related Instructions

PACKSSDW, PACKSSWB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

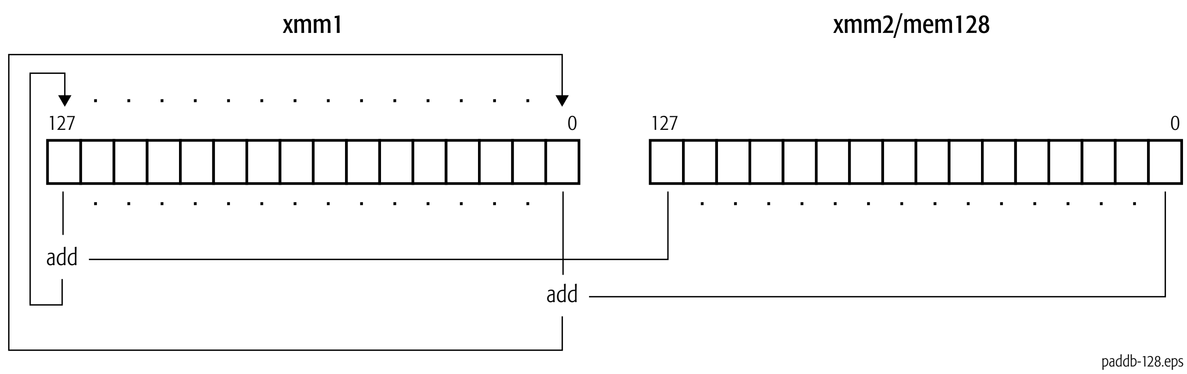
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDB Packed Add Bytes

Adds each packed 8-bit integer value in the first source operand to the corresponding packed 8-bit integer in the second source operand and writes the integer result of each addition in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PADDB *xmm1*, *xmm2/mem128*	66 0F FC /*r*	Adds packed byte integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 8 bits of each result are written in the destination.

Related Instructions

PADDD, PADDQ, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

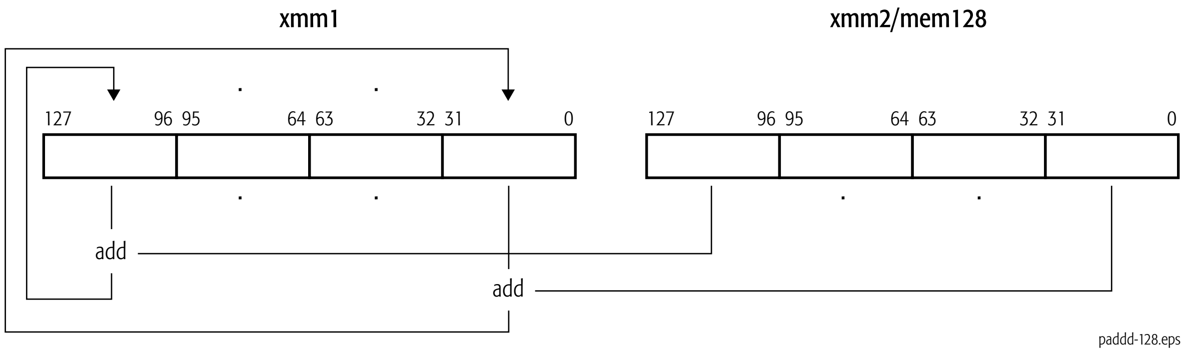
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDD Packed Add Doublewords

Adds each packed 32-bit integer value in the first source operand to the corresponding packed 32-bit integer in the second source operand and writes the integer result of each addition in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PADDD *xmm1*, *xmm2/mem128*	66 0F FE */r*	Adds packed 32-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 32 bits of each result are written in the destination.

Related Instructions

PADDB, PADDQ, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

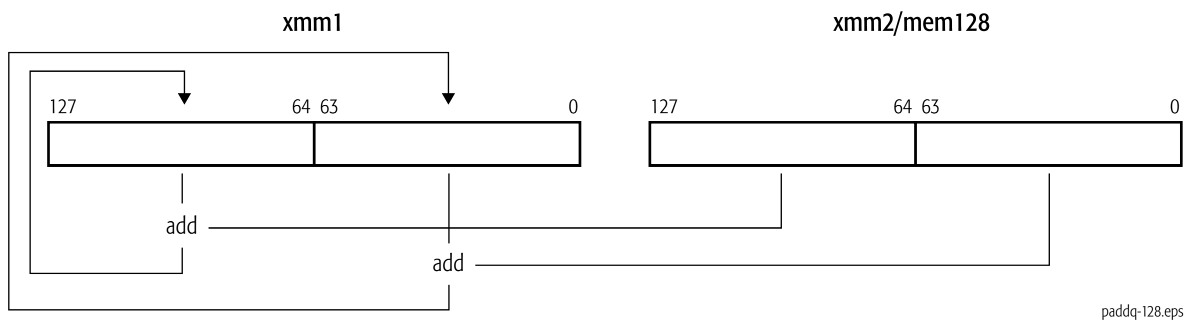
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDQ Packed Add Quadwords

Adds each packed 64-bit integer value in the first source operand to the corresponding packed 64-bit integer in the second source operand and writes the integer result of each addition in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PADDQ *xmm1*, *xmm2/mem128*	66 0F D4 /*r*	Adds packed 64-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 64 bits of each result are written in the destination.

Related Instructions

PADDB, PADDD, PADDSB, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

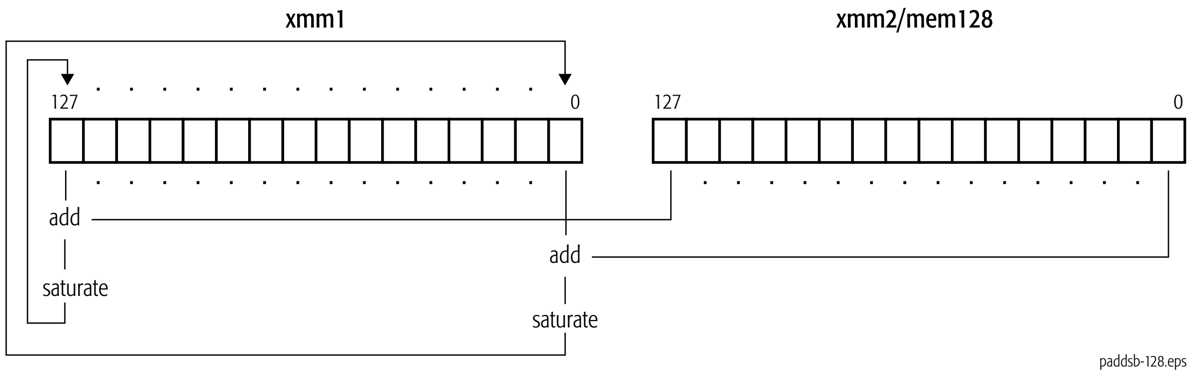
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDSB Packed Add Signed with Saturation Bytes

Adds each packed 8-bit signed integer value in the first source operand to the corresponding packed 8-bit signed integer in the second source operand and writes the signed integer result of each addition in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDSB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PADDSB *xmm1*, *xmm2/mem128*	66 0F EC /*r*	Adds packed byte signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest representable signed 8-bit integer, it is saturated to 7Fh, and if the value is smaller than the smallest signed 8-bit integer, it is saturated to 80h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSW, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

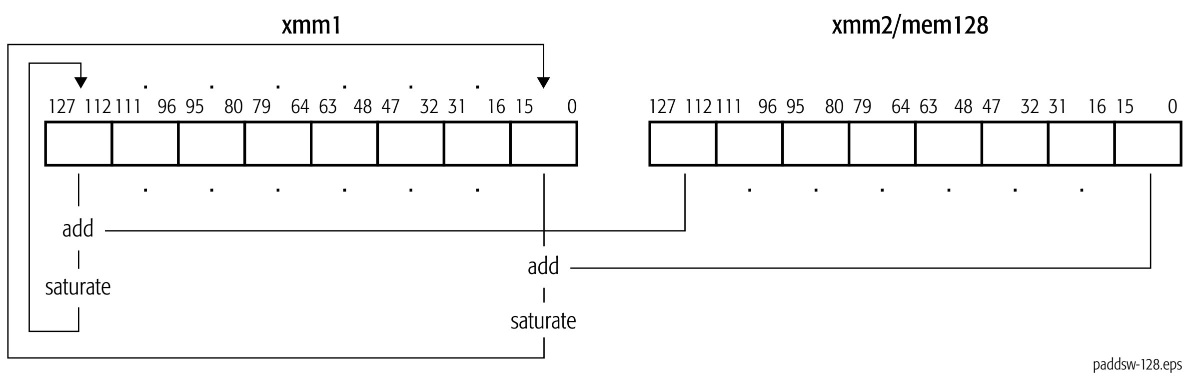
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDSW Packed Add Signed with Saturation Words

Adds each packed 16-bit signed integer value in the first source operand to the corresponding packed 16-bit signed integer in the second source operand and writes the signed integer result of each addition in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDSW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PADDSW *xmm1*, *xmm2/mem128*	66 0F ED /r	Adds packed 16-bit signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest representable signed 16-bit integer, it is saturated to 7FFFh, and if the value is smaller than the smallest signed 16-bit integer, it is saturated to 8000h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDUSB, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

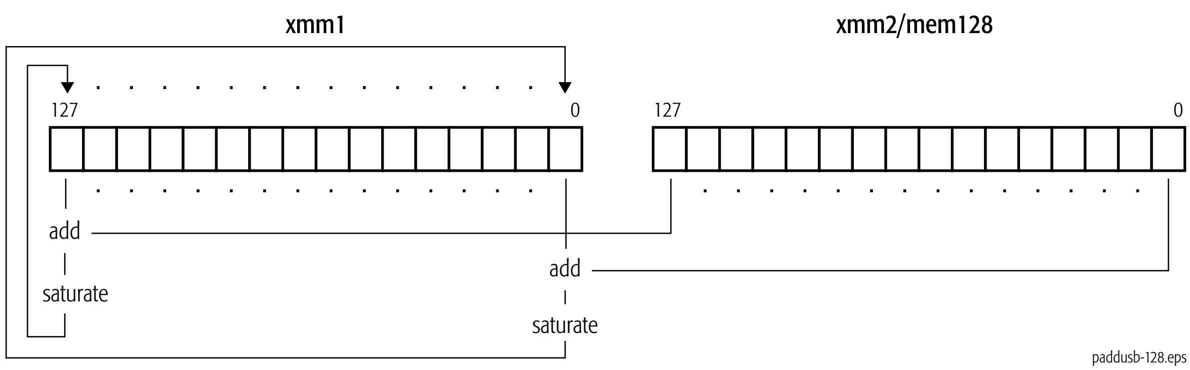
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDUSB Packed Add Unsigned with Saturation Bytes

Adds each packed 8-bit unsigned integer value in the first source operand to the corresponding packed 8-bit unsigned integer in the second source operand and writes the unsigned integer result of each addition in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDUSB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PADDUSB *xmm1*, *xmm2/mem128*	66 0F DC /*r*	Adds packed byte unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest unsigned 8-bit integer, it is saturated to FFh, and if the value is smaller than the smallest unsigned 8-bit integer, it is saturated to 00h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDSW, PADDUSW, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

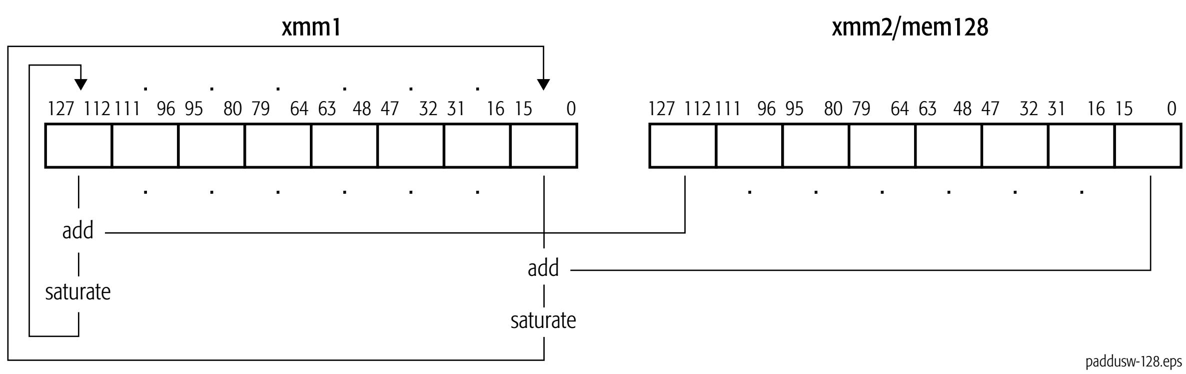
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDUSW Packed Add Unsigned with Saturation Words

Adds each packed 16-bit unsigned integer value in the first source operand to the corresponding packed 16-bit unsigned integer in the second source operand and writes the unsigned integer result of each addition in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDUSW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PADDUSW *xmm1*, *xmm2/mem128*	66 0F DD /*r*	Adds packed 16-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest unsigned 16-bit integer, it is saturated to FFFFh, and if the value is smaller than the smallest unsigned 16-bit integer, it is saturated to 0000h.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDSW, PADDUSB, PADDW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

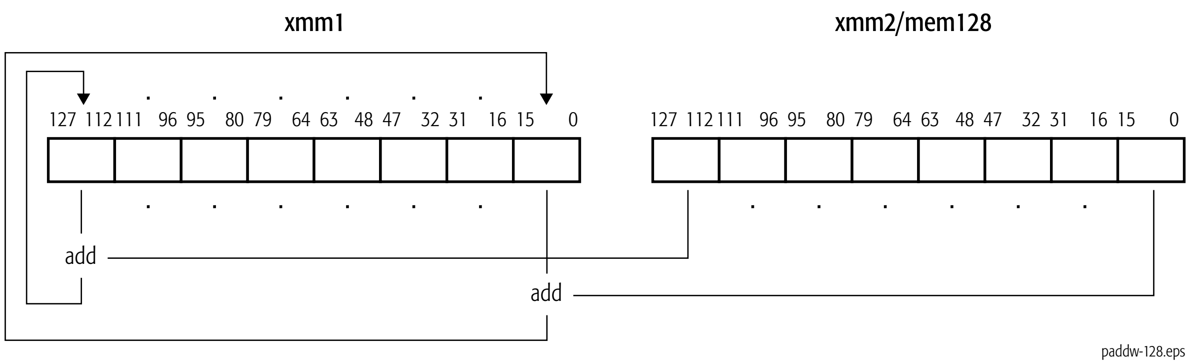
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PADDW Packed Add Words

Adds each packed 16-bit integer value in the first source operand to the corresponding packed 16-bit integer in the second source operand and writes the integer result of each addition in the corresponding word of the destination (second source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PADDW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PADDW *xmm1*, *xmm2/mem128*	66 0F FD /*r*	Adds packed 16-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 16 bits of the result are written in the destination.

Related Instructions

PADDB, PADDD, PADDQ, PADDSB, PADDSW, PADDUSB, PADDUSW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

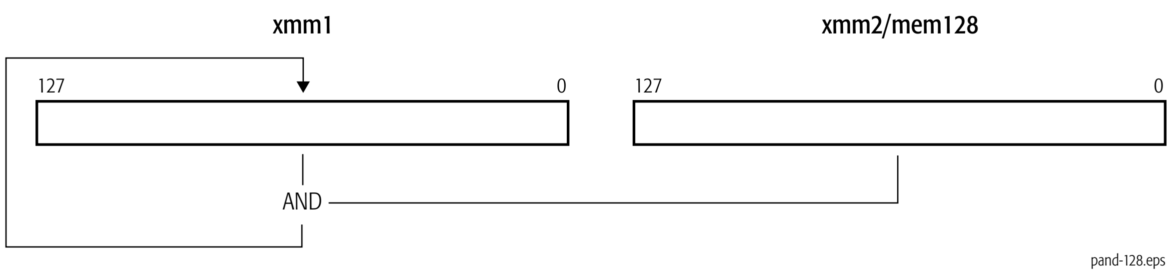
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PAND Packed Logical Bitwise AND

Performs a bitwise logical AND of the values in the first and second source operands and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PAND instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PAND *xmm1*, *xmm2/mem128*	66 0F DB /*r*	Performs bitwise logical AND of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

PANDN, POR, PXOR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

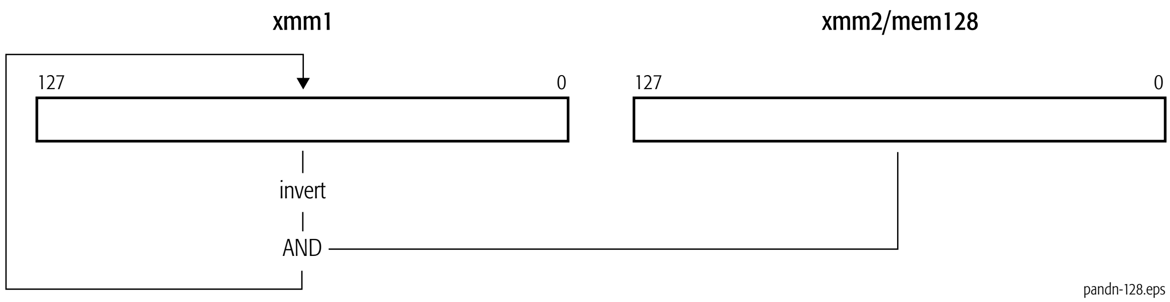
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PANDN Packed Logical Bitwise AND NOT

Performs a bitwise logical AND of the value in the second source operand and the one's complement of the value in the first source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PANDN instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PANDN *xmm1*, *xmm2/mem128*	66 0F DF /*r*	Performs bitwise logical AND NOT of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

PAND, POR, PXOR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

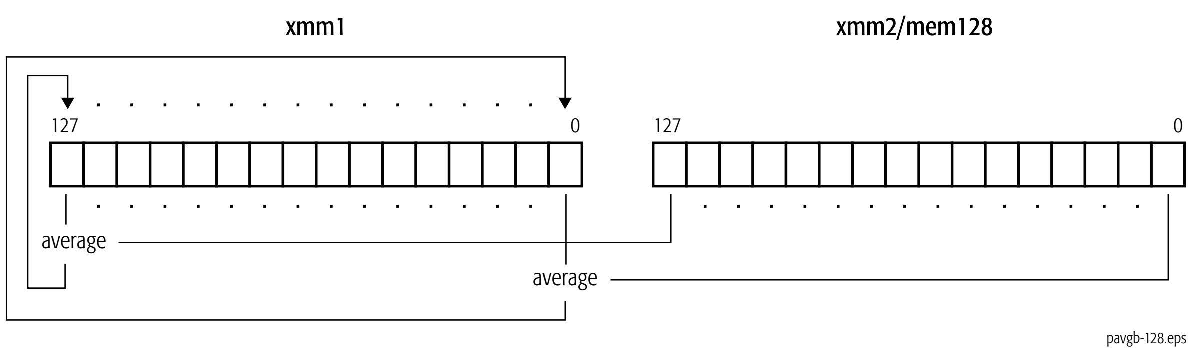
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PAVGB — Packed Average Unsigned Bytes

Computes the rounded average of each packed unsigned 8-bit integer value in the first source operand and the corresponding packed 8-bit unsigned integer in the second source operand and writes each average in the corresponding byte of the destination (first source). The average is computed by adding each pair of operands, adding 1 to the 9-bit temporary sum, and then right-shifting the temporary sum by one bit position. The destination and source operands are an XMM register and another XMM register or 128-bit memory location.

The PAVGB instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PAVGB *xmm1*, *xmm2/mem128*	66 0F E0 /*r*	Averages packed 8-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

PAVGW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

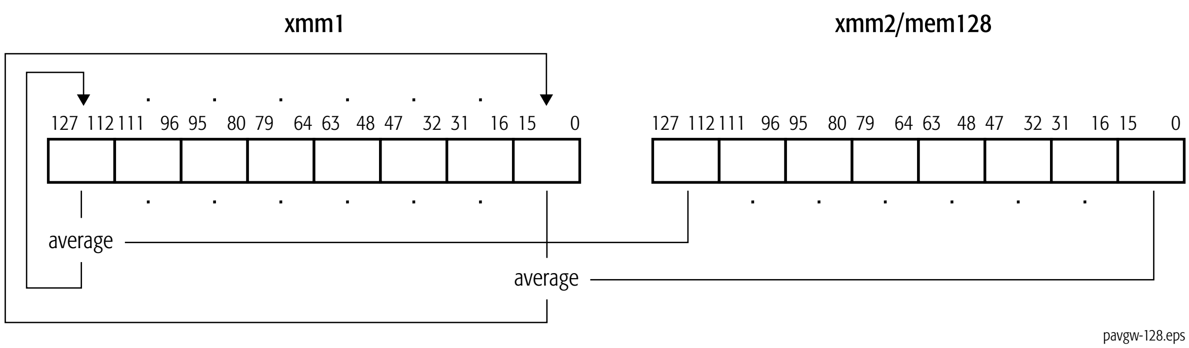
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PAVGW Packed Average Unsigned Words

Computes the rounded average of each packed unsigned 16-bit integer value in the first source operand and the corresponding packed 16-bit unsigned integer in the second source operand and writes each average in the corresponding word of the destination (first source). The average is computed by adding each pair of operands, adding 1 to the 17-bit temporary sum, and then right-shifting the temporary sum by one bit position. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PAVGW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PAVGW *xmm1*, *xmm2/mem128*	66 0F E3 */r*	Averages packed 16-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

PAVGB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

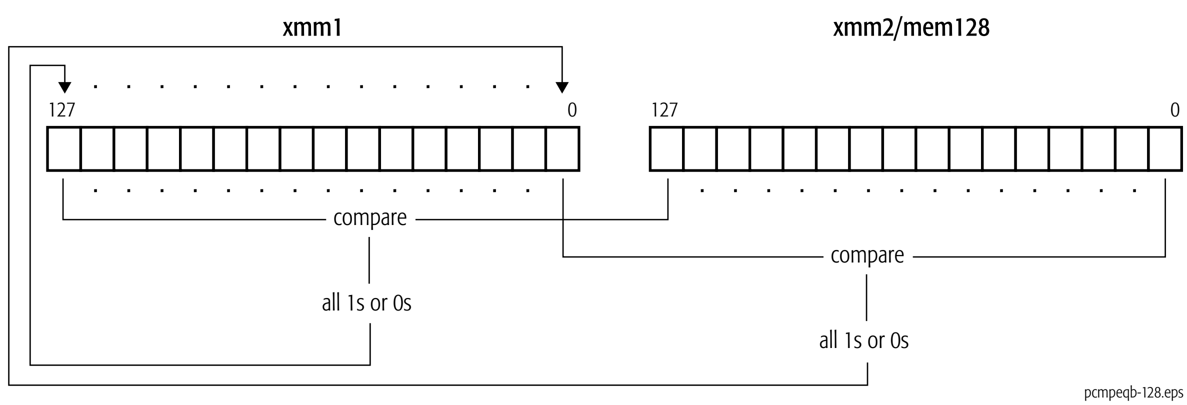
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPEQB Packed Compare Equal Bytes

Compares corresponding packed bytes in the first and second source operands and writes the result of each comparison in the corresponding byte of the destination (first source). For each pair of bytes, if the values are equal, the result is all 1s. If the values are not equal, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PCMPEQB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PCMPEQB *xmm1*, *xmm2/mem128*	66 0F 74 /*r*	Compares packed bytes in an XMM register and an XMM register or 128-bit memory location.



Related Instructions

PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

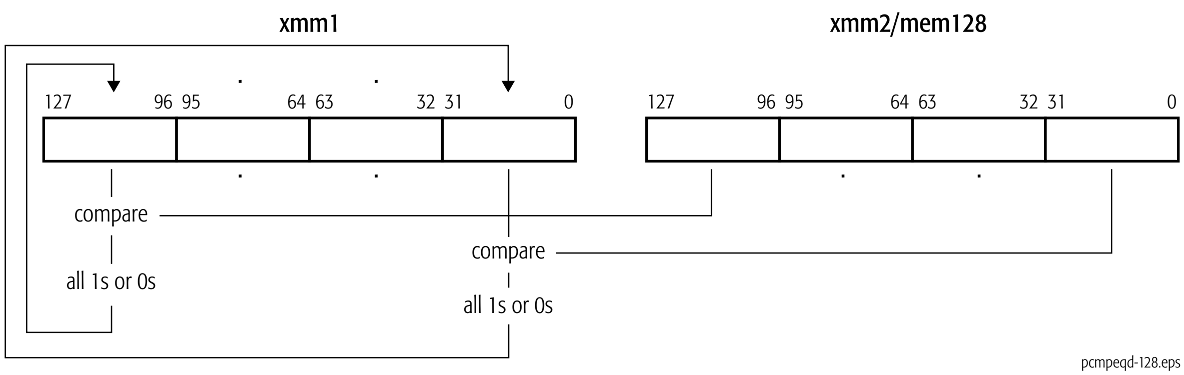
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPEQD Packed Compare Equal Doublewords

Compares corresponding packed 32-bit values in the first and second source operands and writes the result of each comparison in the corresponding 32 bits of the destination (first source). For each pair of doublewords, if the values are equal, the result is all 1s. If the values are not equal, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PCMPEQD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PCMPEQD *xmm1*, *xmm2/mem128*	66 0F 76 /*r*	Compares packed doublewords in an XMM register and an XMM register or 128-bit memory location.



Related Instructions

PCMPEQB, PCMPEQW, PCMPGTB, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

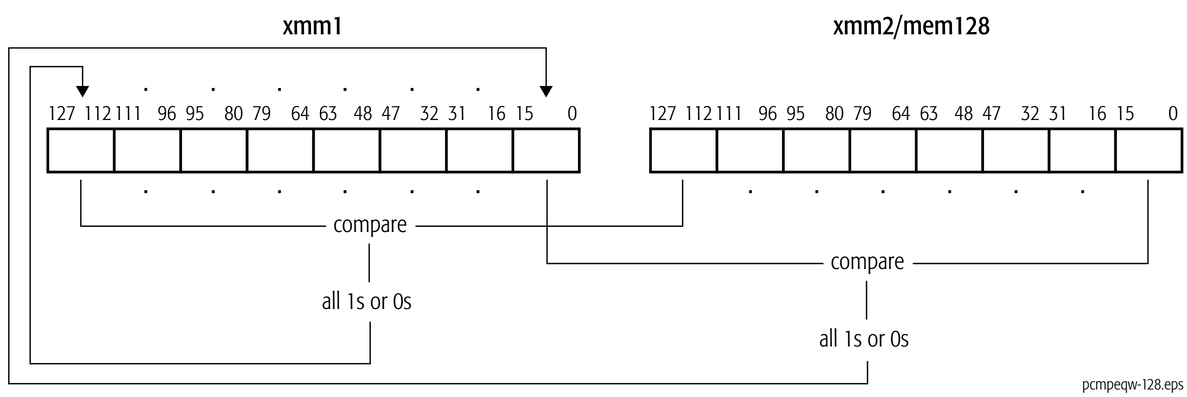
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPEQW Packed Compare Equal Words

Compares corresponding packed 16-bit values in the first and second source operands and writes the result of each comparison in the corresponding 16 bits of the destination (first source). For each pair of words, if the values are equal, the result is all 1s. If the values are not equal, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PCMPEQW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PCMPEQW *xmm1*, *xmm2/mem128*	66 0F 75 /*r*	Compares packed 16-bit values in an XMM register and an XMM register or 128-bit memory location.



Related Instructions

PCMPEQB, PCMPEQD, PCMPGTB, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

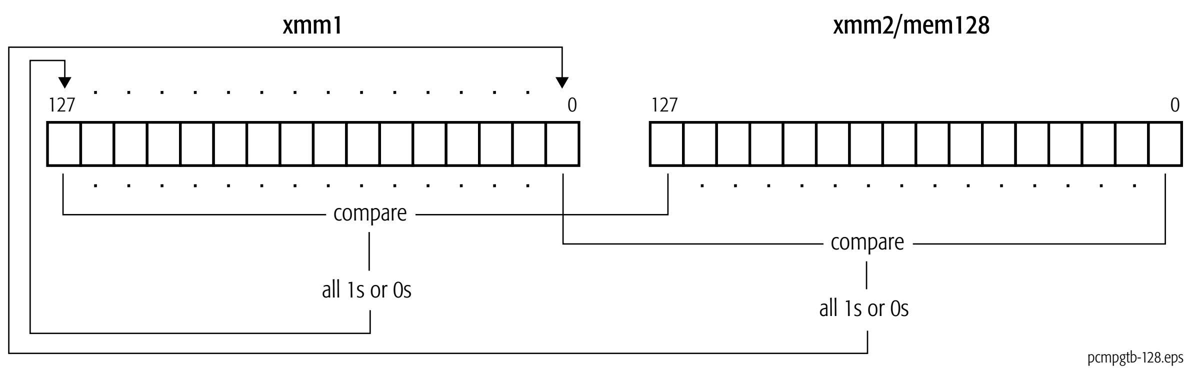
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPGTB Packed Compare Greater Than Signed Bytes

Compares corresponding packed signed bytes in the first and second source operands and writes the result of each comparison in the corresponding byte of the destination (first source). For each pair of bytes, if the value in the first source operand is greater than the value in the second source operand, the result is all 1s. If the value in the first source operand is less than or equal to the value in the second source operand, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PCMPGTB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PCMPGTB *xmm1*, *xmm2/mem128*	66 0F 64 */r*	Compares packed signed bytes in an XMM register and an XMM register or 128-bit memory location.



Related Instructions

PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTD, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

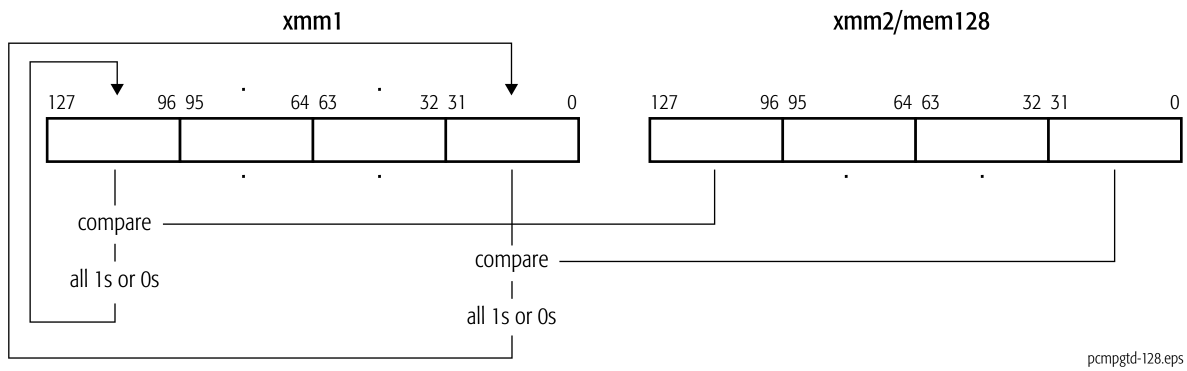
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPGTD Packed Compare Greater Than Signed Doublewords

Compares corresponding packed signed 32-bit values in the first and second source operands and writes the result of each comparison in the corresponding 32 bits of the destination (first source). For each pair of doublewords, if the value in the first source operand is greater than the value in the second source operand, the result is all 1s. If the value in the first source operand is less than or equal to the value in the second source operand, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PCMPGTD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PCMPGTD *xmm1*, *xmm2/mem128*	66 0F 66 /*r*	Compares packed signed 32-bit values in an XMM register and an XMM register or 128-bit memory location.



Related Instructions

PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

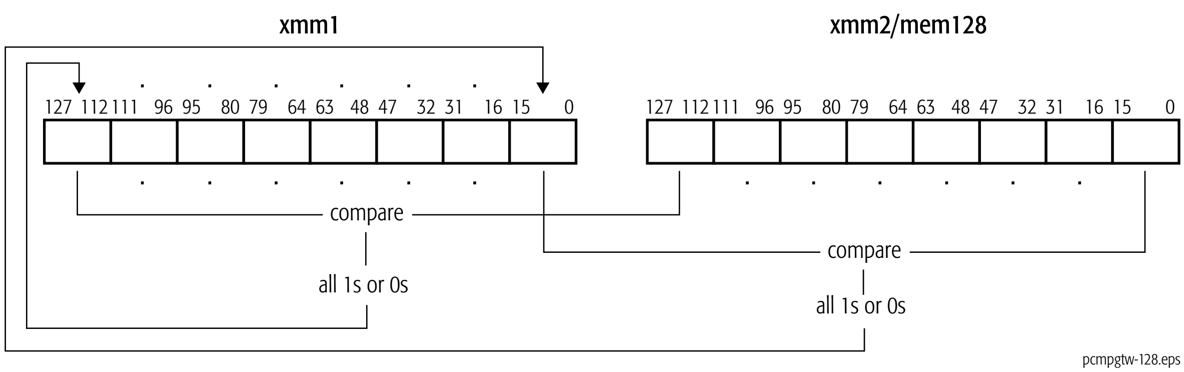
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PCMPGTW — Packed Compare Greater Than Signed Words

Compares corresponding packed signed 16-bit values in the first and second source operands and writes the result of each comparison in the corresponding 16 bits of the destination (first source). For each pair of words, if the value in the first source operand is greater than the value in the second source operand, the result is all 1s. If the value in the first source operand is less than or equal to the value in the second source operand, the result is all 0s. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PCMPGTW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PCMPGTW *xmm1*, *xmm2/mem128*	66 0F 65 /*r*	Compares packed signed 16-bit values in an XMM register and an XMM register or 128-bit memory location.



Related Instructions

PCMPEQB, PCMPEQD, PCMPEQW, PCMPGTB, PCMPGTD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

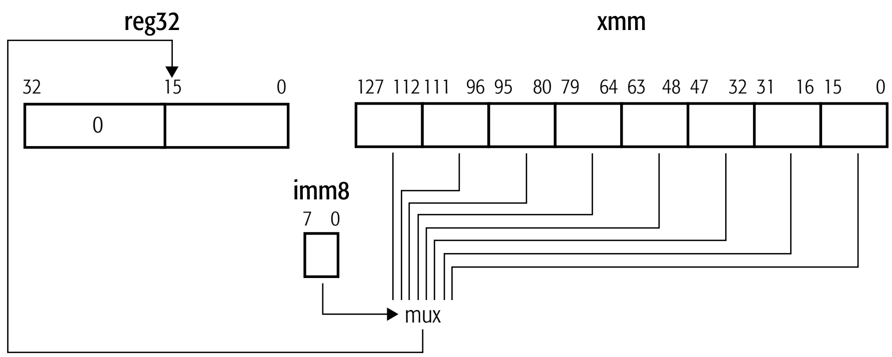
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PEXTRW Extract Packed Word

Extracts a 16-bit value from an XMM register, as selected by the immediate byte operand (as shown in Table 1-2) and writes it to the low-order word of a 32-bit general-purpose register, with zero-extension to 32 bits.

The PEXTRW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PEXTRW *reg32*, *xmm*, *imm8*	66 0F C5 */r ib*	Extracts a 16-bit value from an XMM register and writes it to low-order 16 bits of a general-purpose register.



pextrw-128.eps

Table 1-2. Immediate-Byte Operand Encoding for 128-Bit PEXTRW

Immediate-Byte Bit Field	Value of Bit Field	Source Bits Extracted
2–0	0	15–0
	1	31–16
	2	47–32
	3	63–48
	4	79–64
	5	95–80
	6	111–96
	7	127–112

Related Instructions

PINSRW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

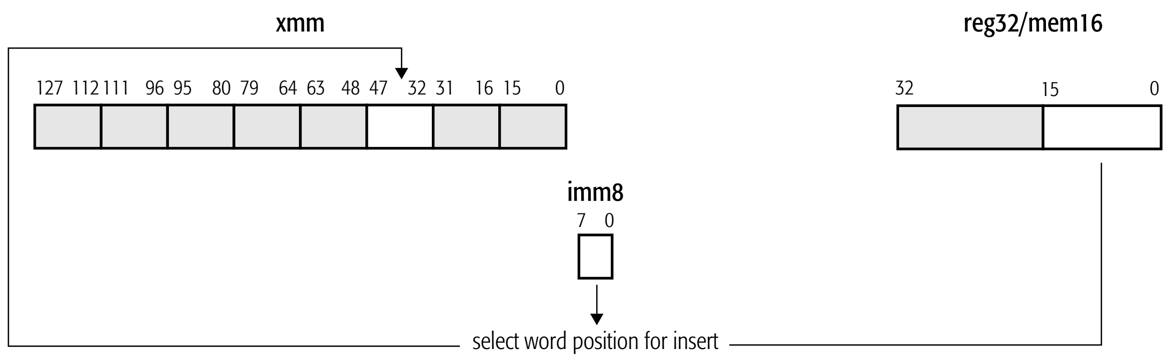
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

PINSRW Packed Insert Word

Inserts a 16-bit value from the low-order word of a 32-bit general purpose register or a 16-bit memory location into an XMM register. The location in the destination register is selected by the immediate byte operand, as shown in Table 1-3 on page 275. The other words in the destination register operand are not modified.

The PINSRW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PINSRW *xmm*, *reg32/mem16*, *imm8*	66 0F C4 /*r ib*	Inserts a 16-bit value from a general-purpose register or memory location into an XMM register.



pinsrw-128.eps

Table 1-3. Immediate-Byte Operand Encoding for 128-Bit PINSRW

Immediate-Byte Bit Field	Value of Bit Field	Destination Bits Filled
2–0	0	15–0
	1	31–16
	2	47–32
	3	63–48
	4	79–64
	5	95–80
	6	111–96
	7	127–112

Related Instructions

PEXTRW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.

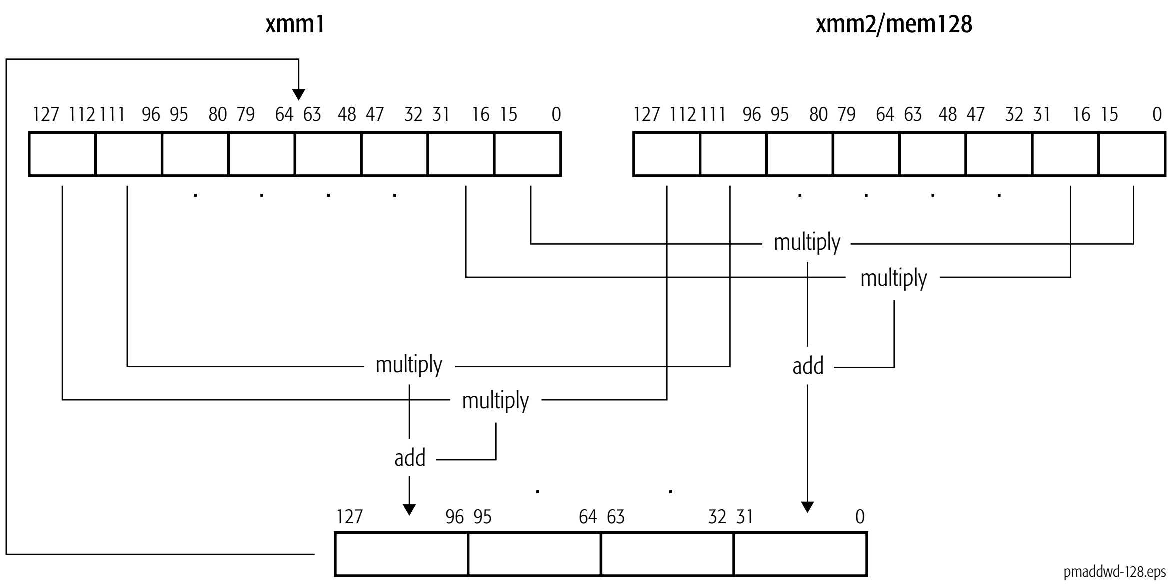
Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

PMADDWD Packed Multiply Words and Add Doublewords

Multiplies each packed 16-bit signed value in the first source operand by the corresponding packed 16-bit signed value in the second source operand, adds the adjacent intermediate 32-bit results of each multiplication (for example, the multiplication results for the adjacent bit fields 63–48 and 47–32, and 31–16 and 15–0), and writes the 32-bit result of each addition in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PMADDWD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMADDWD *xmm1*, *xmm2/mem128*	66 0F F5 /*r*	Multiplies eight packed 16-bit signed values in an XMM register and another XMM register or 128-bit memory location, adds intermediate results, and writes the result in the destination XMM register.



There is only one case in which the result of the multiplication and addition will not fit in a signed 32-bit destination. If all four of the 16-bit source operands used to produce a 32-bit multiply-add result have the value 8000h, the 32-bit result is 8000_0000h, which is incorrect.

Related Instructions

PMULHUW, PMULHW, PMULLW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

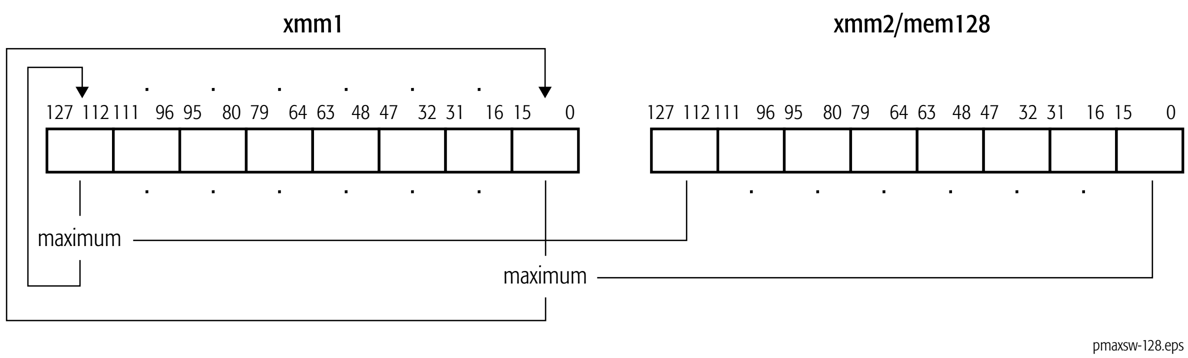
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMAXSW **Packed Maximum Signed Words**

Compares each of the packed 16-bit signed integer values in the first source operand with the corresponding packed 16-bit signed integer value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding word of the destination (first source). The first source/destination and second source operands are an XMM register and an XMM register or 128-bit memory location.

The PMAXSW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMAXSW *xmm1*, *xmm2/mem128*	66 0F EE /*r*	Compares packed signed 16-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the greater value of each comparison in destination XMM register.



Related Instructions

PMAXUB, PMINSW, PMINUB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

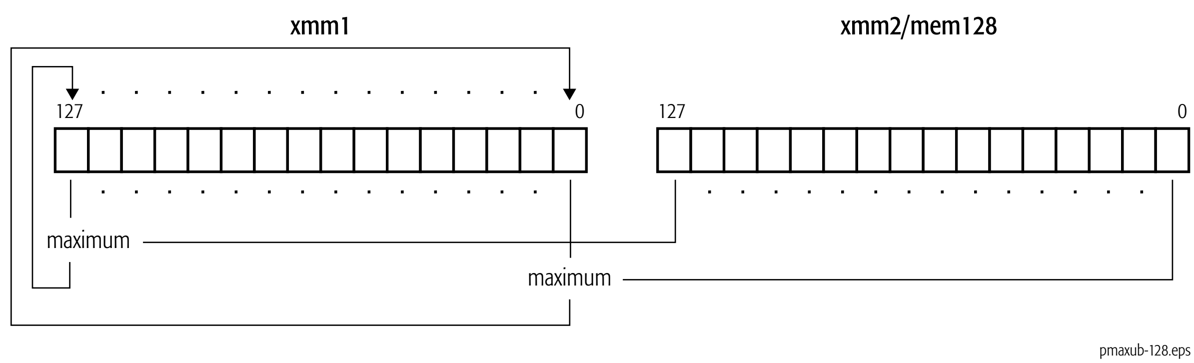
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	A memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMAXUB Packed Maximum Unsigned Bytes

Compares each of the packed 8-bit unsigned integer values in the first source operand with the corresponding packed 8-bit unsigned integer value in the second source operand and writes the numerically greater of the two values for each comparison in the corresponding byte of the destination (first source). The first source/destination and second source operands are an XMM register and an XMM register or 128-bit memory location.

The PMAXUB instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMAXUB *xmm1*, *xmm2/mem128*	66 0F DE /*r*	Compares packed unsigned 8-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the greater value of each compare in the destination XMM register.



Related Instructions

PMAXSW, PMINSW, PMINUB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

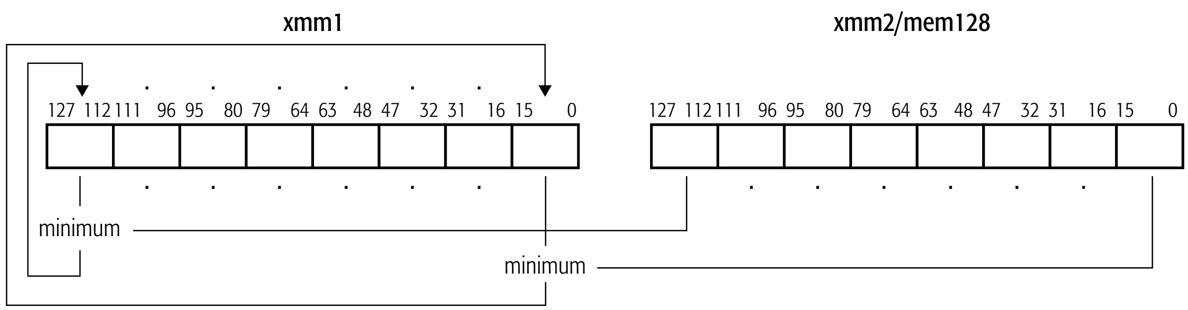
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	A memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMINSW Packed Minimum Signed Words

Compares each of the packed 16-bit signed integer values in the first source operand with the corresponding packed 16-bit signed integer value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding word of the destination (first source). The first source/destination and second source operands are an XMM register and an XMM register or 128-bit memory location.

The PMINSW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMINSW *xmm1*, *xmm2/mem128*	66 0F EA /*r*	Compares packed signed 16-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the lesser value of each compare in the destination XMM register.



pminsw-128.eps

Related Instructions

PMAXSW, PMAXUB, PMINUB

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

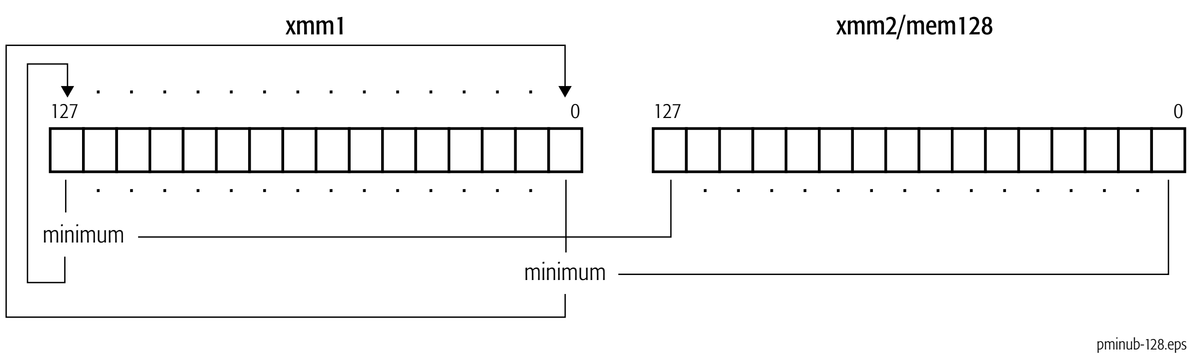
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	A memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMINUB Packed Minimum Unsigned Bytes

Compares each of the packed 8-bit unsigned integer values in the first source operand with the corresponding packed 8-bit unsigned integer value in the second source operand and writes the numerically lesser of the two values for each comparison in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PMINUB instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMINUB *xmm1*, *xmm2/mem128*	66 0F DA /*r*	Compares packed unsigned 8-bit integer values in an XMM register and another XMM register or 128-bit memory location and writes the lesser value of each comparison in the destination XMM register.



Related Instructions

PMAXSW, PMAXUB, PMINSW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

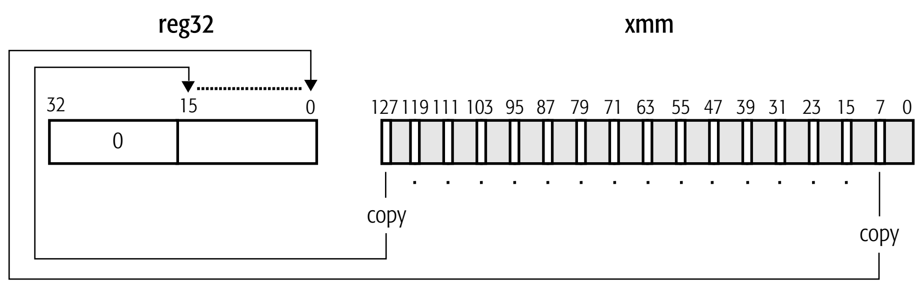
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	A memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMOVMSKB Packed Move Mask Byte

Moves the most-significant bit of each byte in the source operand to the destination, with zero-extension to 32 bits. The destination and source operands are a 32-bit general-purpose register and an XMM register. The result is written to the low-order word of the general-purpose register.

The PMOVMSKB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PMOVMSKB *reg32*, *xmm*	66 0F D7 */r*	Moves most-significant bit of each byte in an XMM register to low-order word of a 32-bit general-purpose register.



pmovmskb-128.eps

Related Instructions

MOVMSKPD, MOVMSKPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

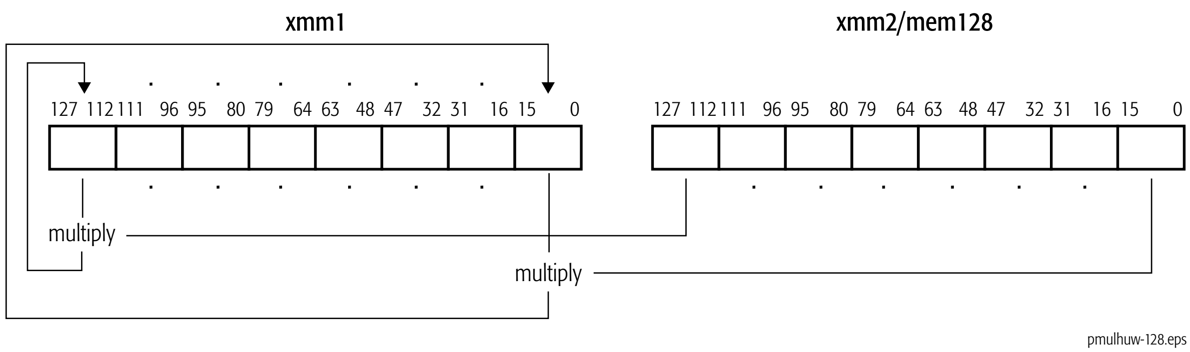
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

PMULHUW Packed Multiply High Unsigned Word

Multiplies each packed unsigned 16-bit values in the first source operand by the corresponding packed unsigned word in the second source operand and writes the high-order 16 bits of each intermediate 32-bit result in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PMULHUW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMULHUW *xmm1*, *xmm2/mem128*	66 0F E4 */r*	Multiplies packed 16-bit values in an XMM register by the packed 16-bit values in another XMM register or 128-bit memory location and writes the high-order 16 bits of each result in the destination XMM register.



Related Instructions

PMADDWD, PMULHW, PMULLW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

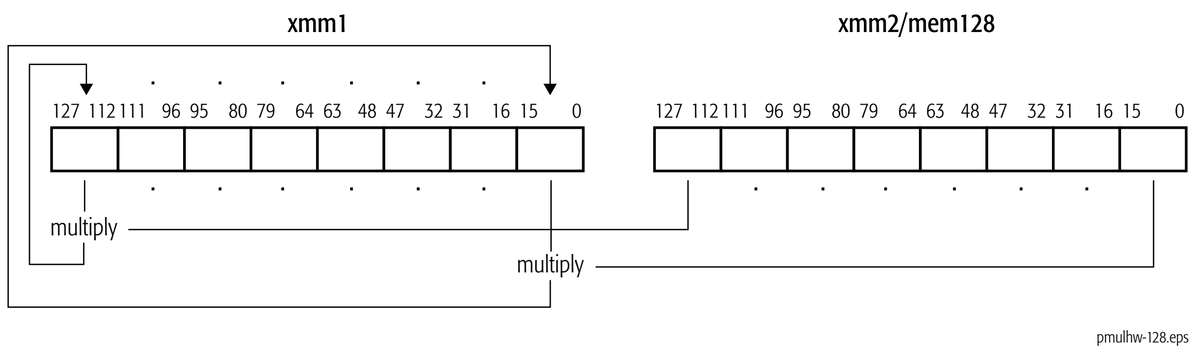
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMULHW Packed Multiply High Signed Word

Multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer in the second source operand and writes the high-order 16 bits of the intermediate 32-bit result of each multiplication in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PMULHW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PMULHW *xmm1*, *xmm2/mem128*	66 0F E5 */r*	Multiplies packed 16-bit signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the high-order 16 bits of each result in the destination XMM register.



Related Instructions

PMADDWD, PMULHUW, PMULLW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

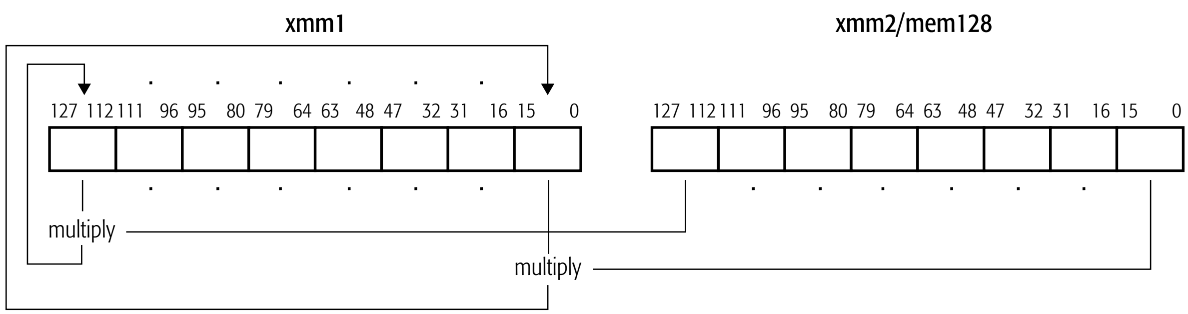
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMULLW Packed Multiply Low Signed Word

Multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer in the second source operand and writes the low-order 16 bits of the intermediate 32-bit result of each multiplication in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PMULLW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMULLW *xmm1*, *xmm2/mem128*	66 0F D5 /*r*	Multiplies packed 16-bit signed integer values in an XMM register and another XMM register or 128-bit memory location and writes the low-order 16 bits of each result in the destination XMM register.



pmullw-128.eps

Related Instructions

PMADDWD, PMULHUW, PMULHW, PMULUDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

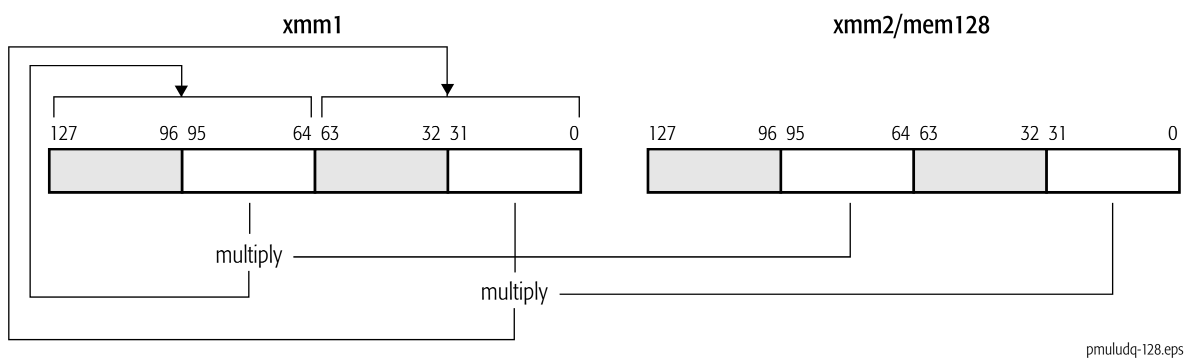
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PMULUDQ Packed Multiply Unsigned Doubleword and Store Quadword

Multiplies two pairs of 32-bit unsigned integer values in the first and second source operands and writes the two 64-bit results in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location. The source operands are in the first (low-order) and third doublewords of the source operands, and the result of each multiply is stored in the first and second quadwords of the destination XMM register.

The PMULUDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PMULUDQ *xmm1*, *xmm2/mem128*	66 0F F4 */r*	Multiplies two pairs of 32-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the two 64-bit results in the destination XMM register.



Related Instructions

PMADDWD, PMULHUW, PMULHW, PMULLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

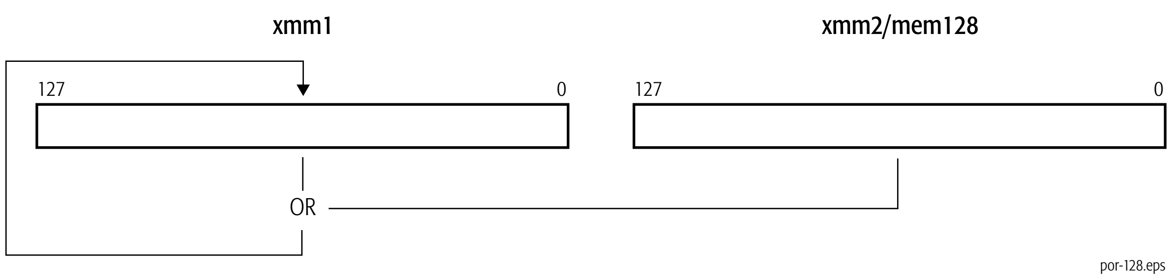
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

POR Packed Logical Bitwise OR

Performs a bitwise logical OR of the values in the first and second source operands and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The POR instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
POR *xmm1*, *xmm2/mem128*	66 0F EB /*r*	Performs bitwise logical OR of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

PAND, PANDN, PXOR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

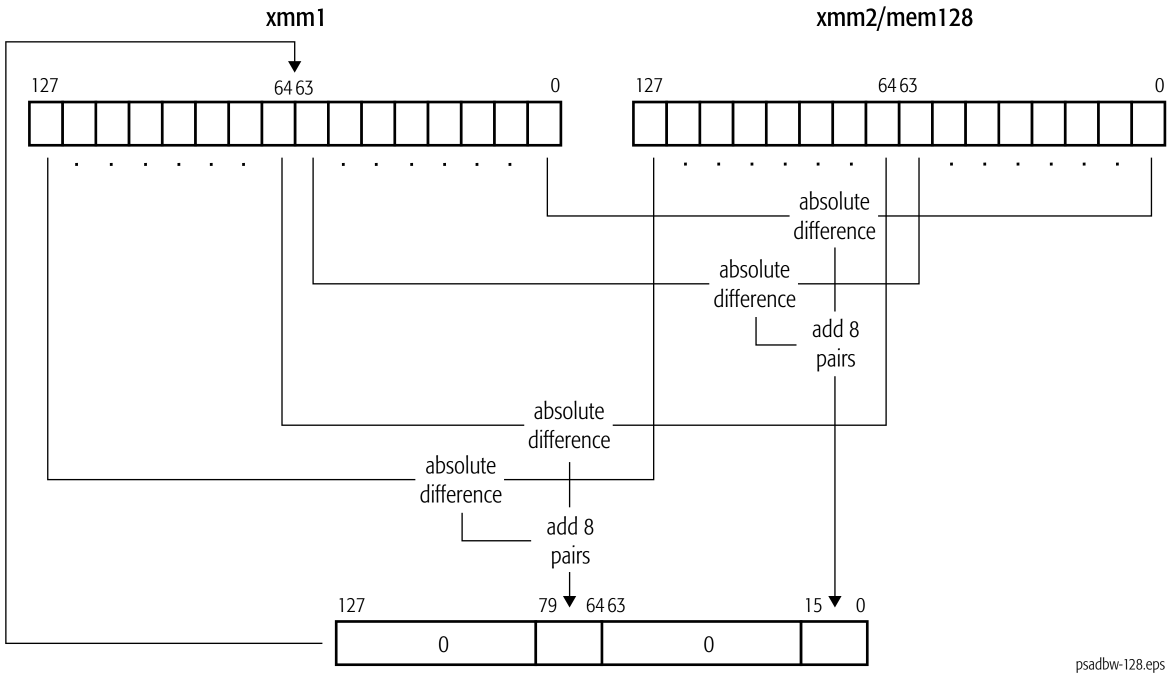
PSADBW Packed Sum of Absolute Differences of Bytes Into a Word

Computes the absolute differences of eight corresponding packed 8-bit unsigned integers in the first and second source operands and writes the unsigned 16-bit integer result of the sum of the eight differences in a word in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The sum of the differences of the eight bytes in the high-order quadwords of the source operands are written in the least-significant word of the high-order quadword in the destination XMM register, with the remaining bytes cleared to all 0s. The sum of the differences of the eight bytes in the low-order quadwords of the source operands are written in the least-significant word of the low-order quadword in the destination XMM register, with the remaining bytes cleared to all 0s.

The PSADBW instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSADBW *xmm1*, *xmm2/mem128*	66 0F F6 */r*	Compute the sum of the absolute differences of two sets of packed 8-bit unsigned integer values in an XMM register and another XMM register or 128-bit memory location and writes the 16-bit unsigned integer result in the destination XMM register.



rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 in CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.

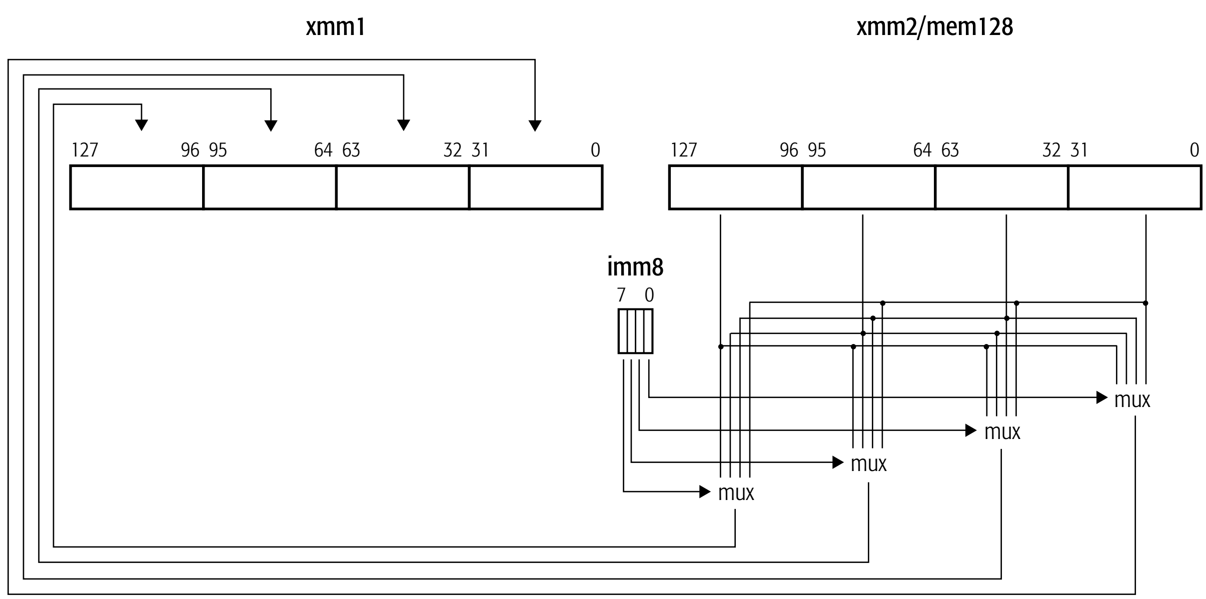
Exception	Real	Virtual 8086	Protected	Cause of Exception
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSHUFD Packed Shuffle Doublewords

Moves any one of the four packed doublewords in an XMM register or 128-bit memory location to each doubleword in another XMM register. In each case, the value of the destination doubleword is determined by a two-bit field in the immediate-byte operand, with bits 0 and 1 selecting the contents of the low-order doubleword, bits 2 and 3 selecting the second doubleword, bits 4 and 5 selecting the third doubleword, and bits 6 and 7 selecting the high-order doubleword. Refer to Table 1-4 on page 303. A doubleword in the source operand may be copied to more than one doubleword in the destination.

The PSHUFD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PSHUFD *xmm1*, *xmm2/mem128*, *imm8*	66 0F 70 */r ib*	Moves packed 32-bit values in an XMM register or 128-bit memory location to doubleword locations in another XMM register, as selected by the immediate-byte operand.



pshufd.eps

Table 1-4. Immediate-Byte Operand Encoding for PSHUFD

Destination Bits Filled	Immediate-Byte Bit Field	Value of Bit Field	Source Bits Moved
31–0	1–0	0	31–0
		1	63–32
		2	95–64
		3	127–96
63–32	3–2	0	31–0
		1	63–32
		2	95–64
		3	127–96
95–64	5–4	0	31–0
		1	63–32
		2	95–64
		3	127–96
127–96	7–6	0	31–0
		1	63–32
		2	95–64
		3	127–96

Related Instructions

PSHUFHW, PSHUFLW, PSHUFW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSHUFHW Packed Shuffle High Words

Moves any one of the four packed words in the high-order quadword of an XMM register or 128-bit memory location to each word in the high-order quadword of another XMM register. In each case, the value of the destination word is determined by a two-bit field in the immediate-byte operand, with bits 0 and 1 selecting the contents of the low-order word, bits 2 and 3 selecting the second word, bits 4 and 5 selecting the third word, and bits 6 and 7 selecting the high-order word. Refer to Table 1-5 on page 306. A word in the source operand may be copied to more than one word in the destination. The low-order quadword of the source operand is copied to the low-order quadword of the destination register.

The PSHUFHW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PSHUFHW *xmm1*, *xmm2/mem128*, *imm8*	F3 0F 70 */r ib*	Shuffles packed 16-bit values in high-order quadword of an XMM register or 128-bit memory location and puts the result in high-order quadword of another XMM register.

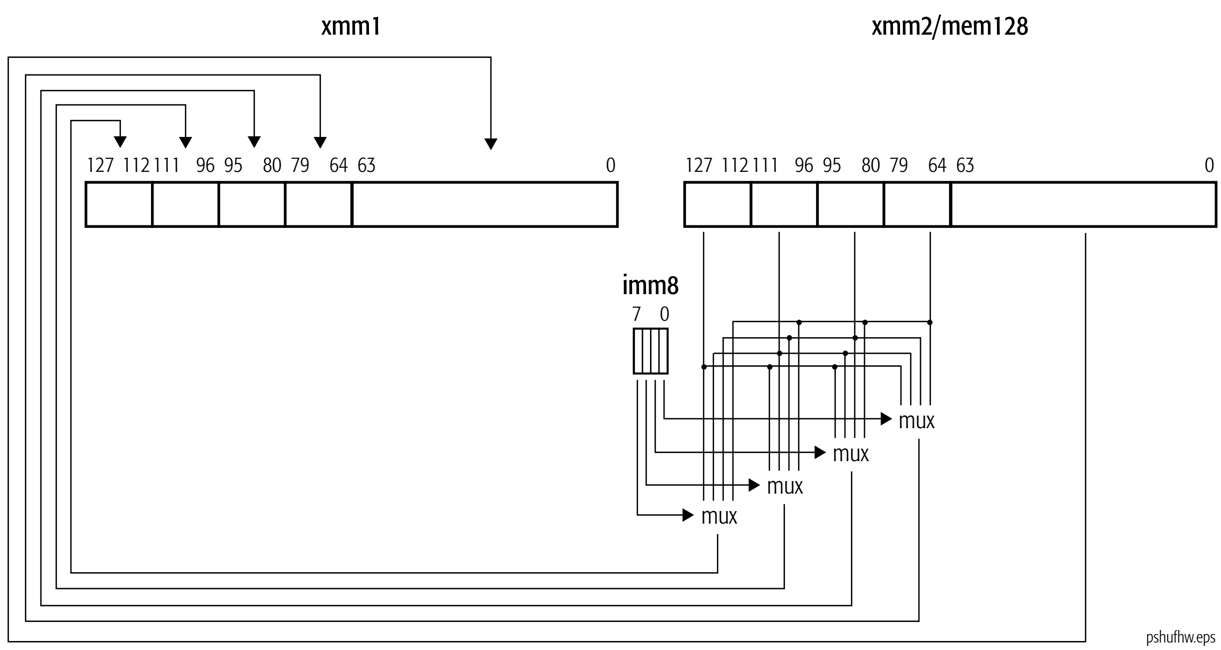


Table 1-5. Immediate-Byte Operand Encoding for PSHUFHW

Destination Bits Filled	Immediate-Byte Bit Field	Value of Bit Field	Source Bits Moved
79–64	1–0	0	79–64
		1	95–80
		2	111–96
		3	127–112
95–80	3–2	0	79–64
		1	95–80
		2	111–96
		3	127–112
111–96	5–4	0	79–64
		1	95–80
		2	111–96
		3	127–112
127–112	7–6	0	79–64
		1	95–80
		2	111–96
		3	127–112

Related Instructions

PSHUFD, PSHUFLW, PSHUFW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSHUFLW — Packed Shuffle Low Words

Moves any one of the four packed words in the low-order quadword of an XMM register or 128-bit memory location to each word in the low-order quadword of another XMM register. In each case, the selection of the value of the destination word is determined by a two-bit field in the immediate-byte operand, with bits 0 and 1 selecting the contents of the low-order word, bits 2 and 3 selecting the second word, bits 4 and 5 selecting the third word, and bits 6 and 7 selecting the high-order word. Refer to Table 1-6 on page 309. A word in the source operand may be copied to more than one word in the destination. The high-order quadword of the source operand is copied to the high-order quadword of the destination register.

The PSHUFLW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSHUFLW *xmm1*, *xmm2/mem128*, *imm8*	F2 0F 70 */r ib*	Shuffles packed 16-bit values in low-order quadword of an XMM register or 128-bit memory location and puts the result in low-order quadword of another XMM register.

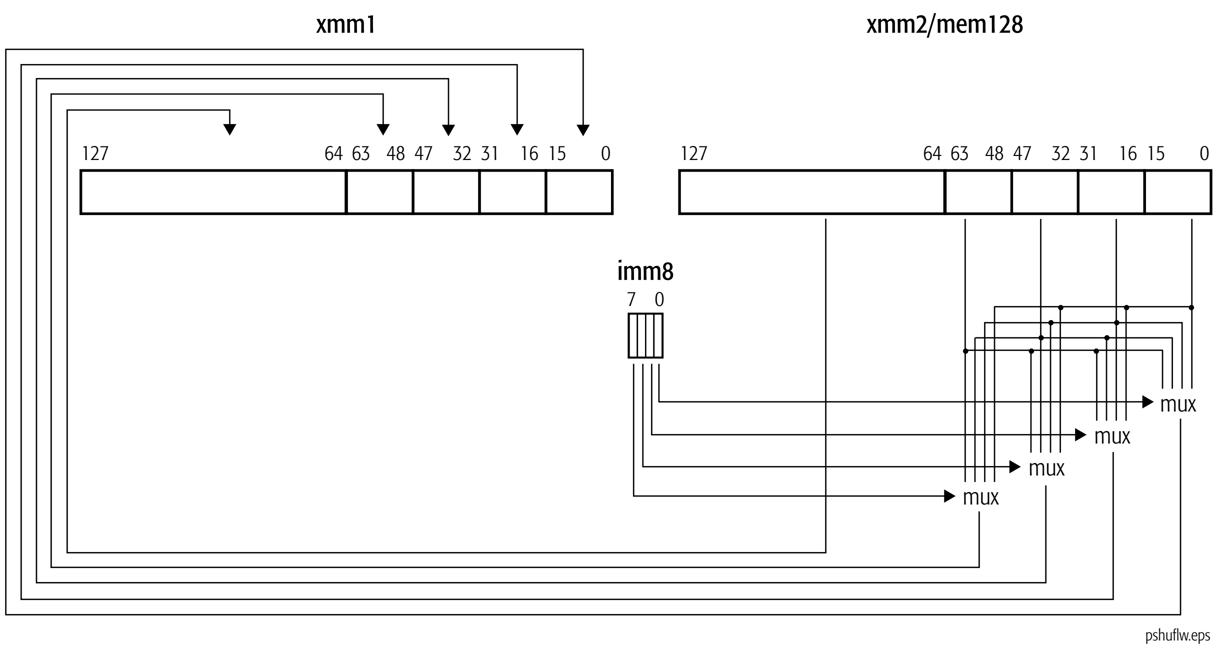


Table 1-6. Immediate-Byte Operand Encoding for PSHUFLW

Destination Bits Filled	Immediate-Byte Bit Field	Value of Bit Field	Source Bits Moved
15–0	1–0	0	15–0
		1	31–16
		2	47–32
		3	63–48
31–16	3–2	0	15–0
		1	31–16
		2	47–32
		3	63–48
47–32	5–4	0	15–0
		1	31–16
		2	47–32
		3	63–48
63–48	7–6	0	15–0
		1	31–16
		2	47–32
		3	63–48

Related Instructions

PSHUFD, PSHUFHW, PSHUFW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSLLD Packed Shift Left Logical Doublewords

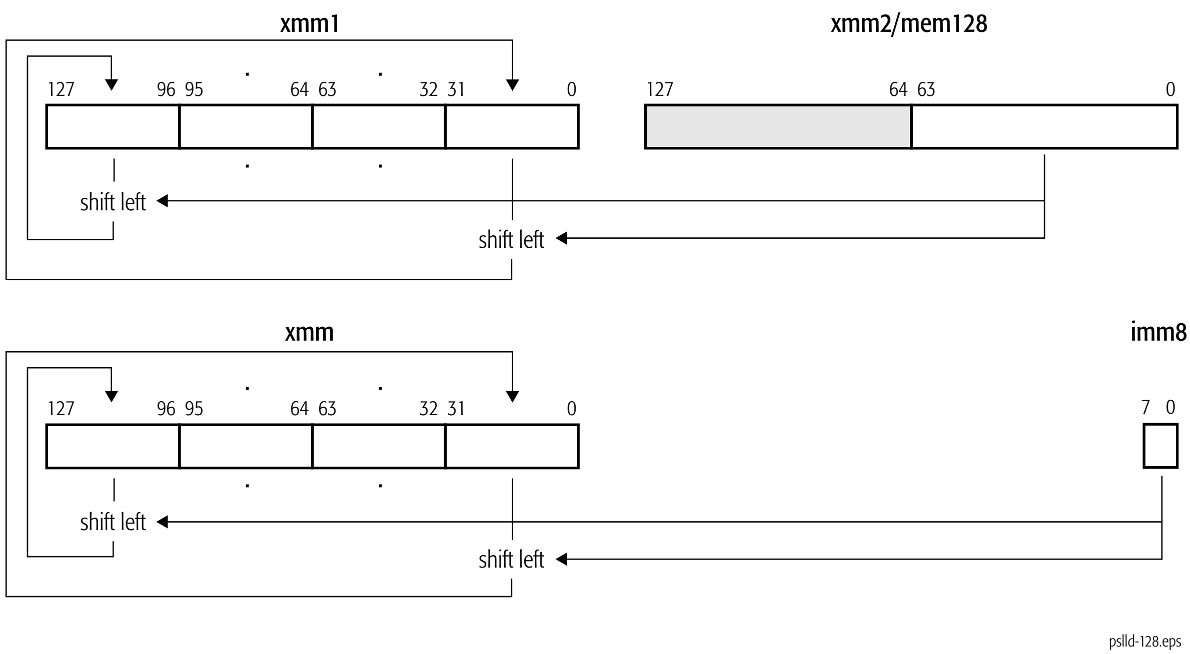
Left-shifts each of the packed 32-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding doubleword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The low-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 31, the destination is cleared to all 0s.

The PSLLD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSLLD *xmm1*, *xmm2/mem128*	66 0F F2 /*r*	Left-shifts packed doublewords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSLLD *xmm*, *imm8*	66 0F 72 /6 *ib*	Left-shifts packed doublewords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

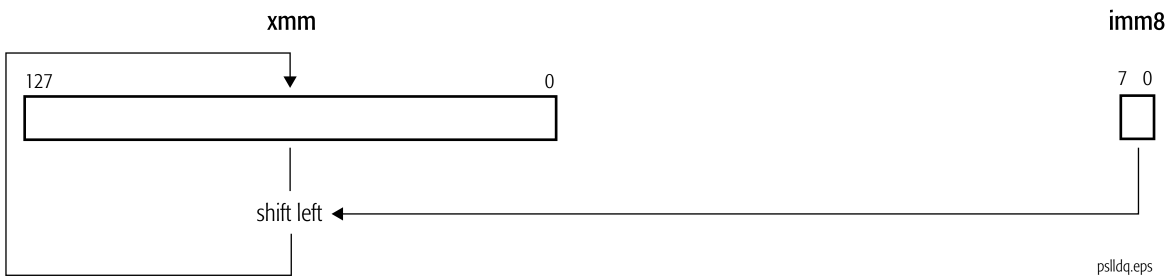
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSLLDQ Packed Shift Left Logical Double Quadword

Left-shifts the 128-bit (double quadword) value in an XMM register by the number of bytes specified in an immediate byte value. The low-order bytes that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination XMM register is cleared to all 0s.

The PSLLDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSLLDQ *xmm*, *imm8*	66 0F 73 /7 *ib*	Left-shifts double quadword value in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

PSLLQ Packed Shift Left Logical Quadwords

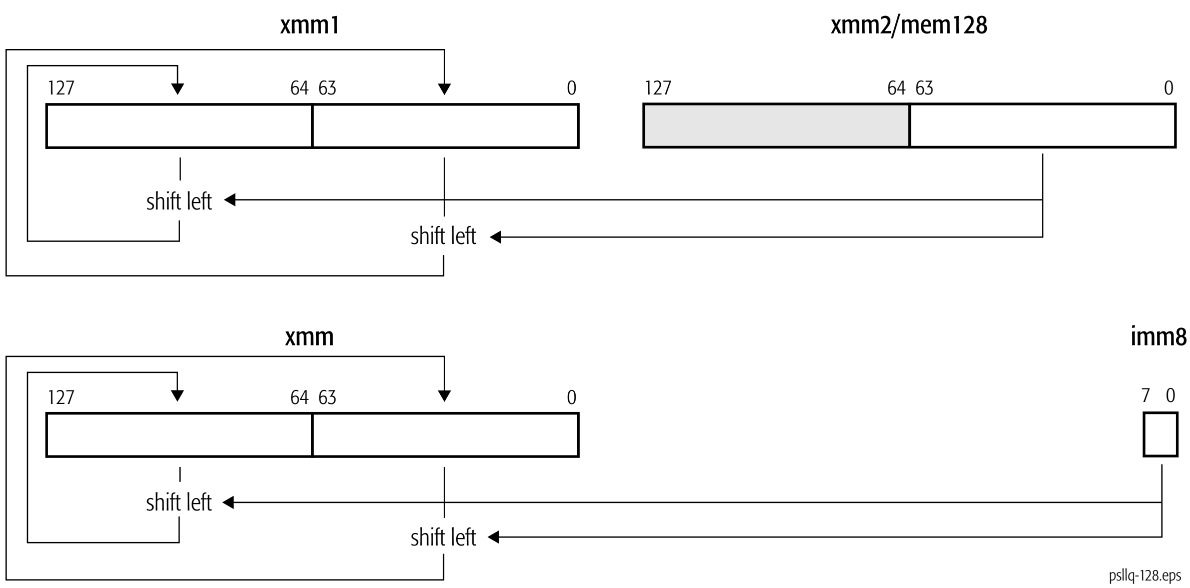
Left-shifts each 64-bit value in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding quadword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The low-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 63, the destination is cleared to all 0s.

The PSLLQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSLLQ *xmm1*, *xmm2/mem128*	66 0F F3 /*r*	Left-shifts packed quadwords in XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSLLQ *xmm*, *imm8*	66 0F 73 /6 *ib*	Left-shifts packed quadwords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSLLW Packed Shift Left Logical Words

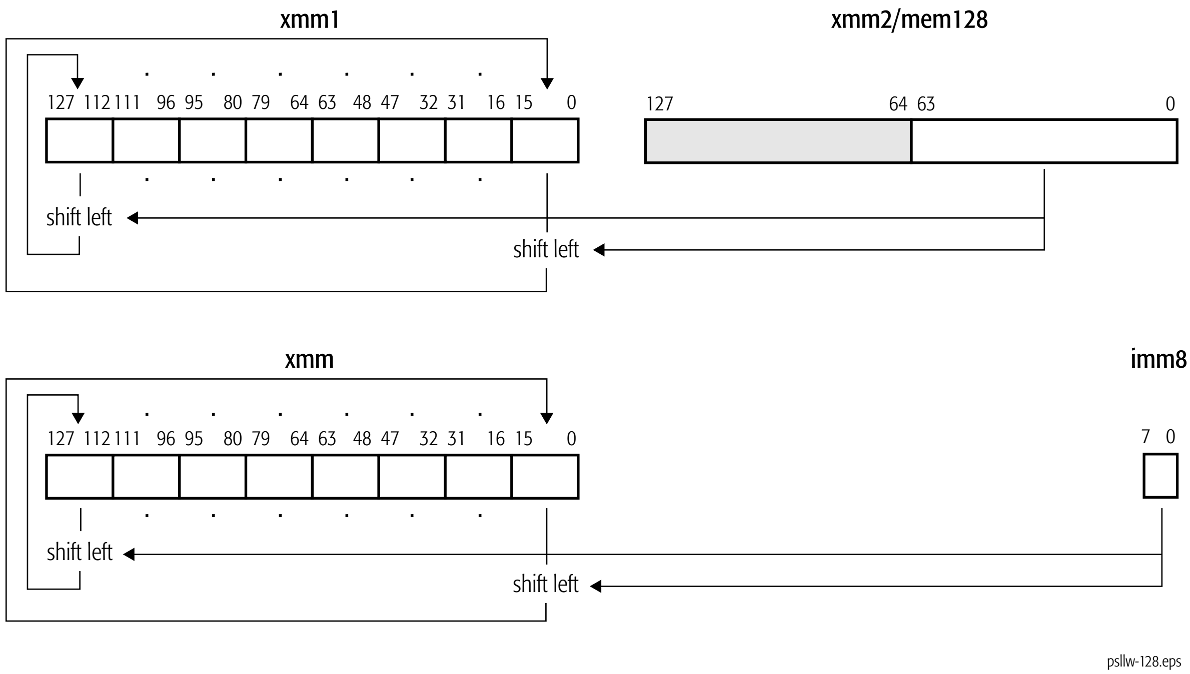
Left-shifts each of the packed 16-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding word of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value

The low-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination is cleared to all 0s.

The PSLLW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSLLW *xmm1*, *xmm2/mem128*	66 0F F1 /*r*	Left-shifts packed words in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSLLW *xmm*, *imm8*	66 0F 71 /6 *ib*	Left-shifts packed words in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSRAD Packed Shift Right Arithmetic Doublewords

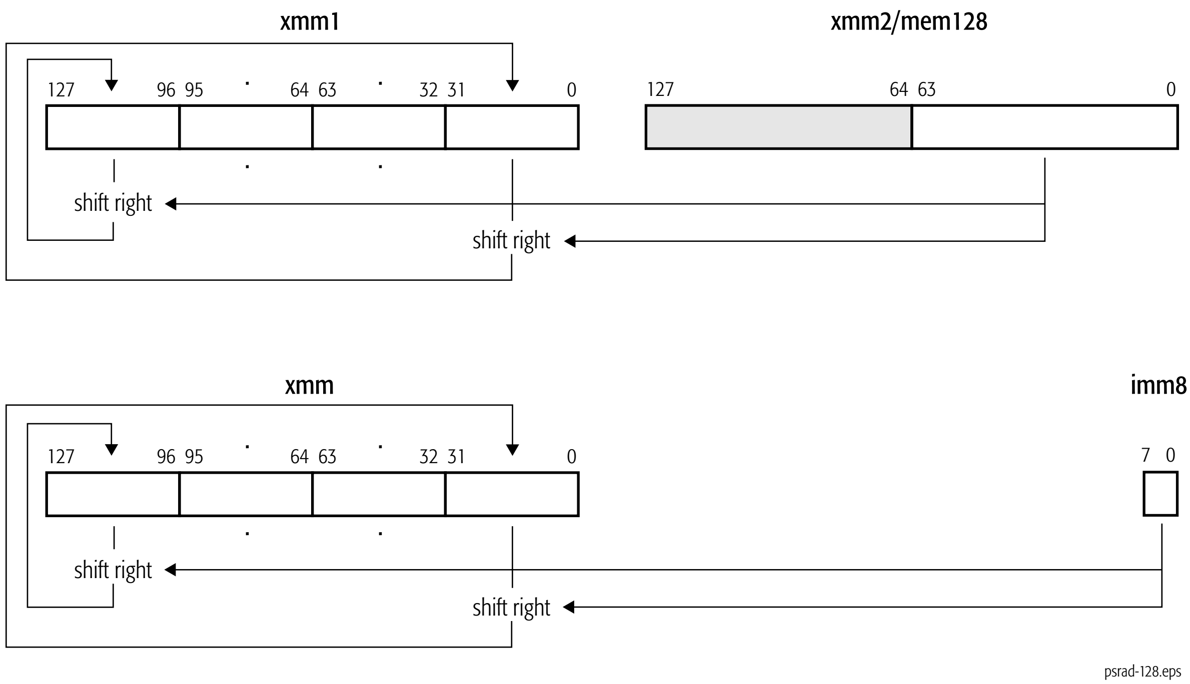
Right-shifts each of the packed 32-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding doubleword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are filled with the sign bit of the doubleword's initial value. If the shift value is greater than 31, each doubleword in the destination is filled with the sign bit of the doubleword's initial value.

The PSRAD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PSRAD *xmm1*, *xmm2/mem128*	66 0F E2 /*r*	Right-shifts packed doublewords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRAD *xmm*, *imm8*	66 0F 72 /4 *ib*	Right-shifts packed doublewords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAW, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSRAW Packed Shift Right Arithmetic Words

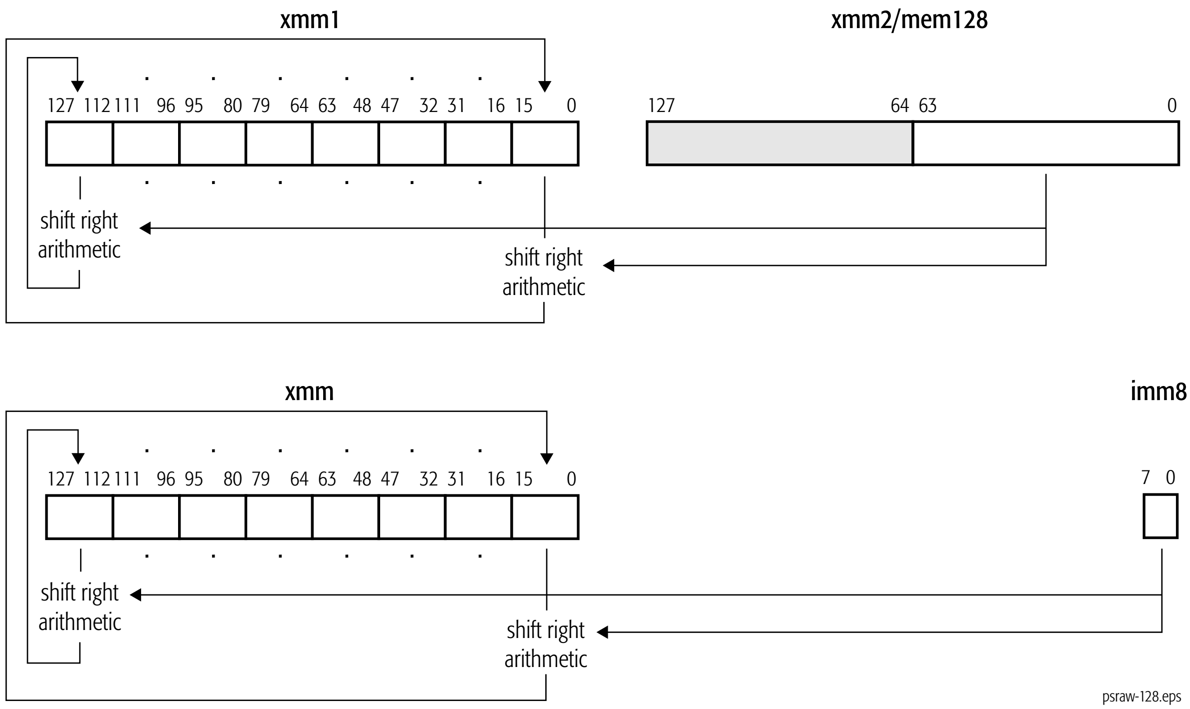
Right-shifts each of the packed 16-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding word of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are filled with the sign bit of the word's initial value. If the shift value is greater than 15, each word in the destination is filled with the sign bit of the word's initial value.

The PSRAW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PSRAW *xmm1*, *xmm2/mem128*	66 0F E1 /*r*	Right-shifts packed words in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRAW *xmm*, *imm8*	66 0F 71 /4 *ib*	Right-shifts packed words in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRLD, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSRLD — Packed Shift Right Logical Doublewords

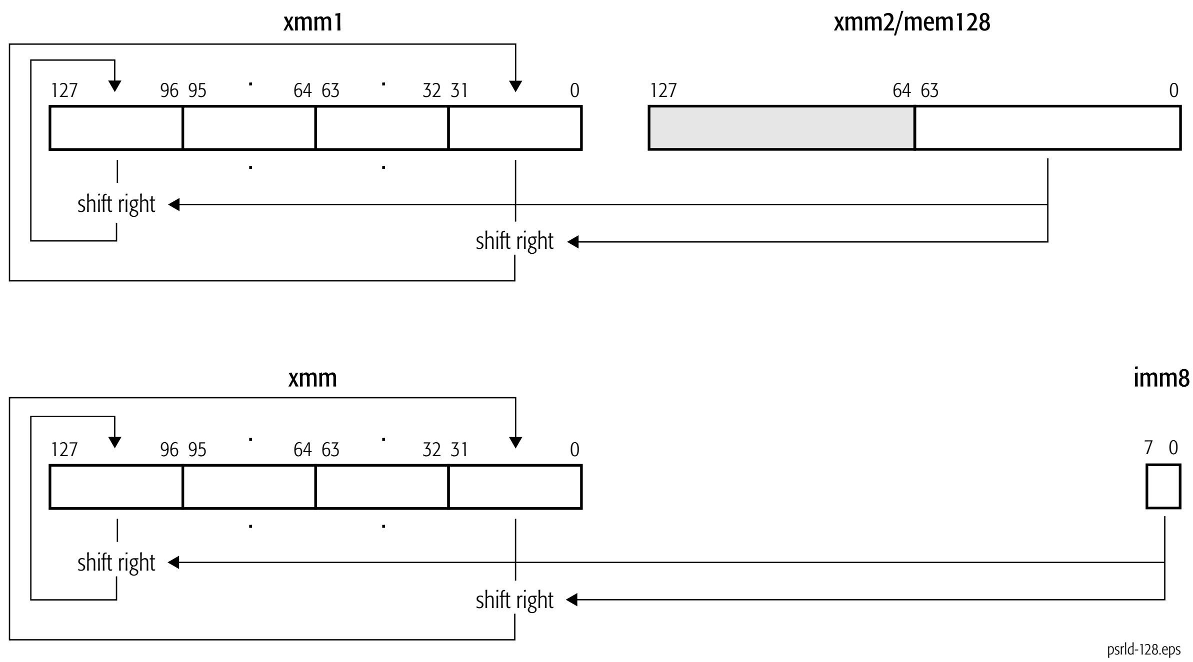
Right-shifts each of the packed 32-bit values in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding doubleword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 31, the destination is cleared to 0.

The PSRLD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	**Opcode**	**Description**
PSRLD *xmm1*, *xmm2/mem128*	66 0F D2 */r*	Right-shifts packed doublewords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRLD *xmm*, *imm8*	66 0F 72 /2 *ib*	Right-shifts packed doublewords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLDQ, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

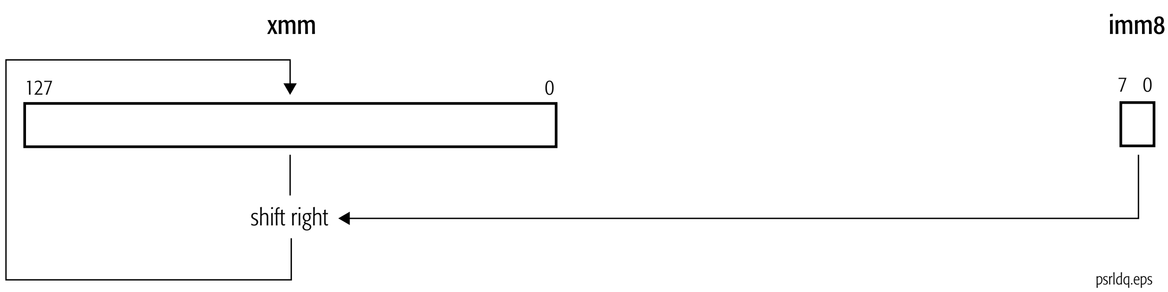
Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSRLDQ Packed Shift Right Logical Double Quadword

Right-shifts the 128-bit (double quadword) value in an XMM register by the number of bytes specified in an immediate byte value. The high-order bytes that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination XMM register is cleared to all 0s.

The PSRLDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSRLDQ *xmm*, *imm8*	66 0F 73 /3 *ib*	Right-shifts double quadword value in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

PSRLQ Packed Shift Right Logical Quadwords

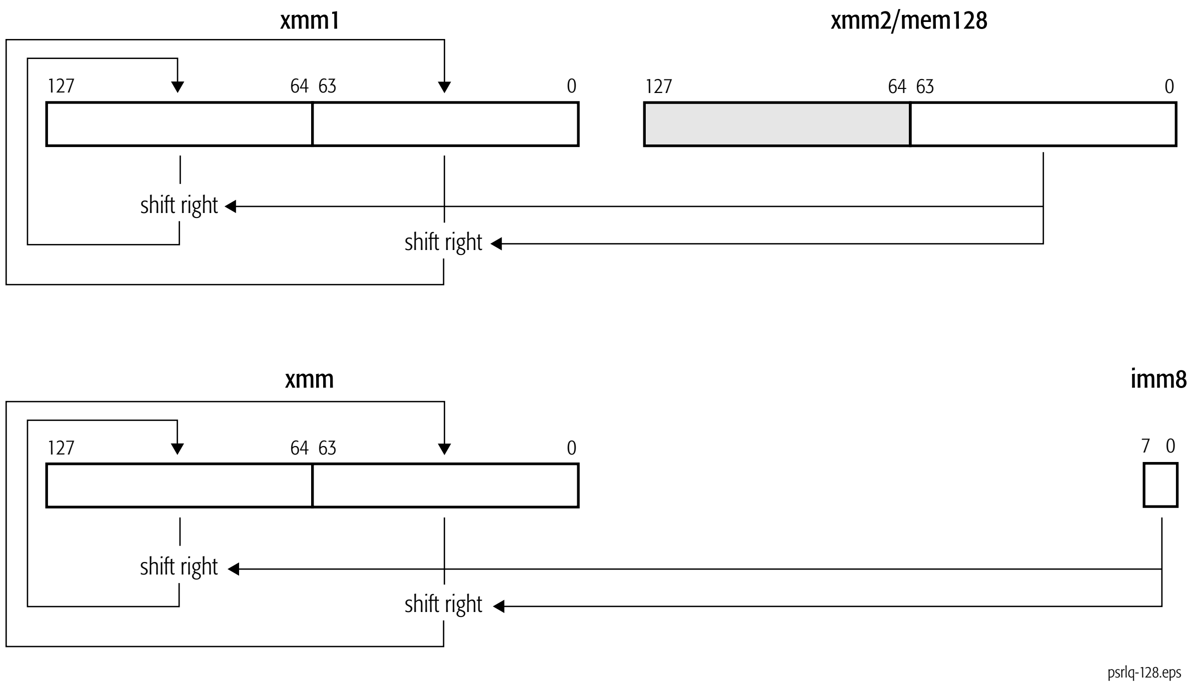
Right-shifts each 64-bit value in the first source operand by the number of bits specified in the second source operand and writes each shifted value in the corresponding quadword of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 63, the destination is cleared to 0.

The PSRLQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PSRLQ *xmm1*, *xmm2/mem128*	66 0F D3 /*r*	Right-shifts packed quadwords in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRLQ *xmm*, *imm8*	66 0F 73 /2 *ib*	Right-shifts packed quadwords in an XMM register by the amount specified in an immediate byte value.



Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSRLW Packed Shift Right Logical Words

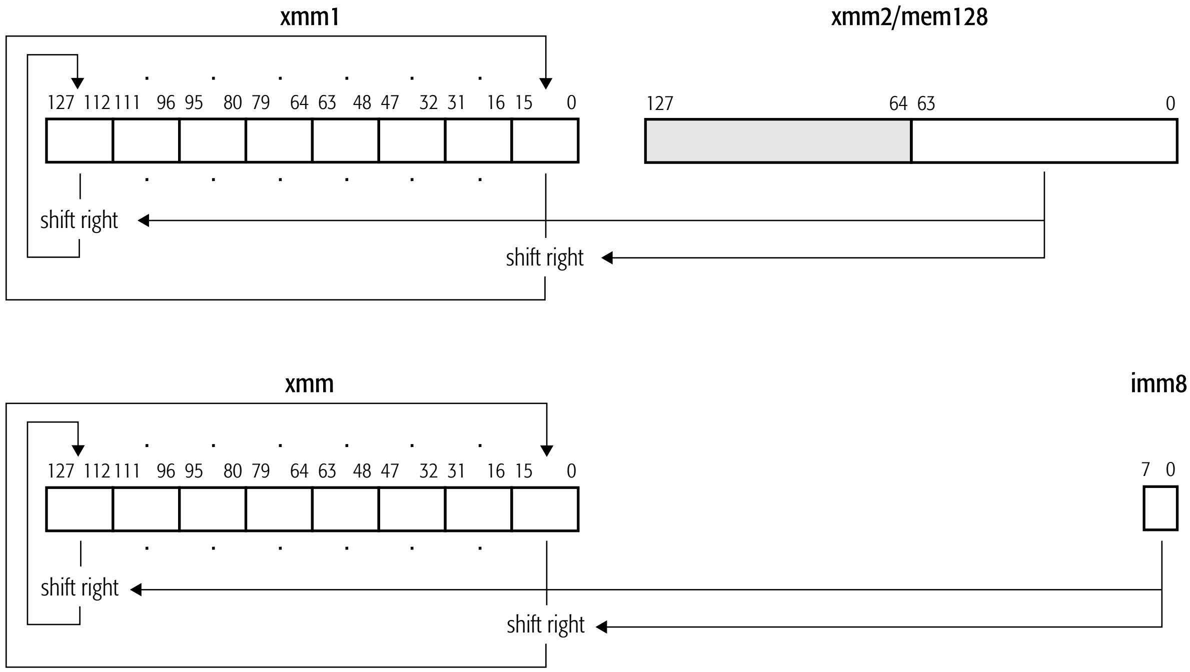
Right-shifts each of the packed 16-bit values in the first source operand by the number of bits specified in the second operand and writes each shifted value in the corresponding word of the destination (first source). The first source/destination and second source operands are:

- an XMM register and another XMM register or 128-bit memory location, or
- an XMM register and an immediate byte value.

The high-order bits that are emptied by the shift operation are cleared to 0. If the shift value is greater than 15, the destination is cleared to 0.

The PSRLW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSRLW *xmm1*, *xmm2/mem128*	66 0F D1 /*r*	Right-shifts packed words in an XMM register by the amount specified in the low 64 bits of an XMM register or 128-bit memory location.
PSRLW *xmm*, *imm8*	66 0F 71 /2 *ib*	Right-shifts packed words in an XMM register by the amount specified in an immediate byte value.



psrlw-128.eps

Related Instructions

PSLLD, PSLLDQ, PSLLQ, PSLLW, PSRAD, PSRAW, PSRLD, PSRLDQ, PSRLQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
	X	X	X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

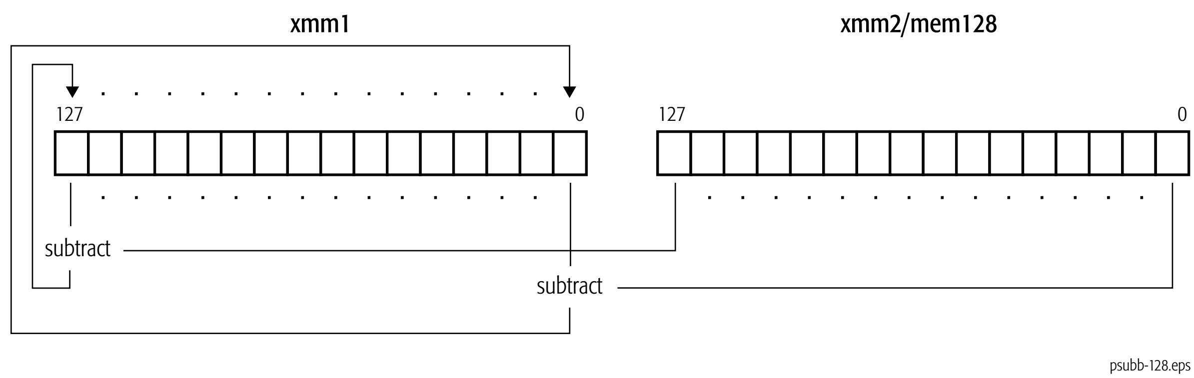
PSUBB Packed Subtract Bytes

Subtracts each packed 8-bit integer value in the second source operand from the corresponding packed 8-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 8 bits of each result are written in the destination.

The PSUBB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBB *xmm1*, *xmm2/mem128*	66 0F F8 /*r*	Subtracts packed byte integer values in an XMM register or 128-bit memory location from packed byte integer values in another XMM register and writes the result in the destination XMM register.



Related Instructions

PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

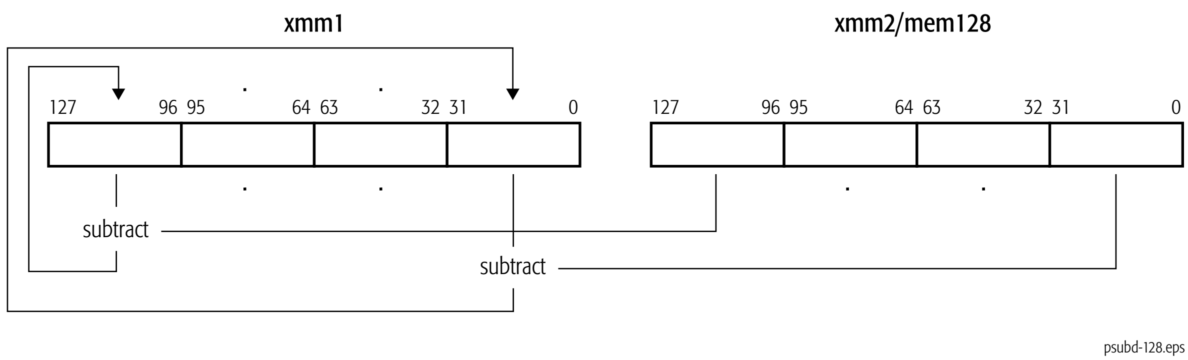
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBD Packed Subtract Doublewords

Subtracts each packed 32-bit integer value in the second source operand from the corresponding packed 32-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PSUBD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBD *xmm1*, *xmm2/mem128*	66 0F FA /*r*	Subtracts packed 32-bit integer values in an XMM register or 128-bit memory location from packed 32-bit integer values in another XMM register and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 32 bits of each result are written in the destination.

Related Instructions

PSUBB, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

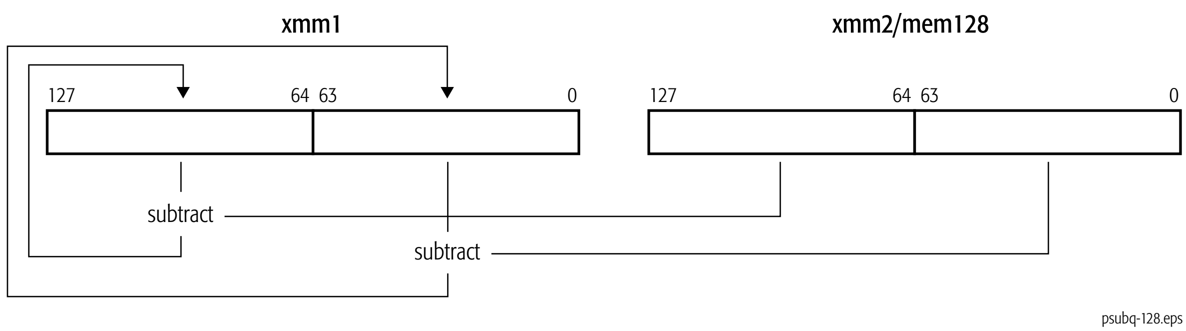
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBQ Packed Subtract Quadword

Subtracts each packed 64-bit integer value in the second source operand from the corresponding packed 64-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding quadword of the destination (first source). The first source/destination and source operands are an XMM register and another XMM register or 128-bit memory location.

The PSUBQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBQ *xmm1*, *xmm2/mem128*	66 0F FB /r	Subtracts packed 64-bit integer values in an XMM register or 128-bit memory location from packed 64-bit integer values in another XMM register and writes the result in the destination XMM register.



This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 64 bits of each result are written in the destination.

Related Instructions

PSUBB, PSUBD, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

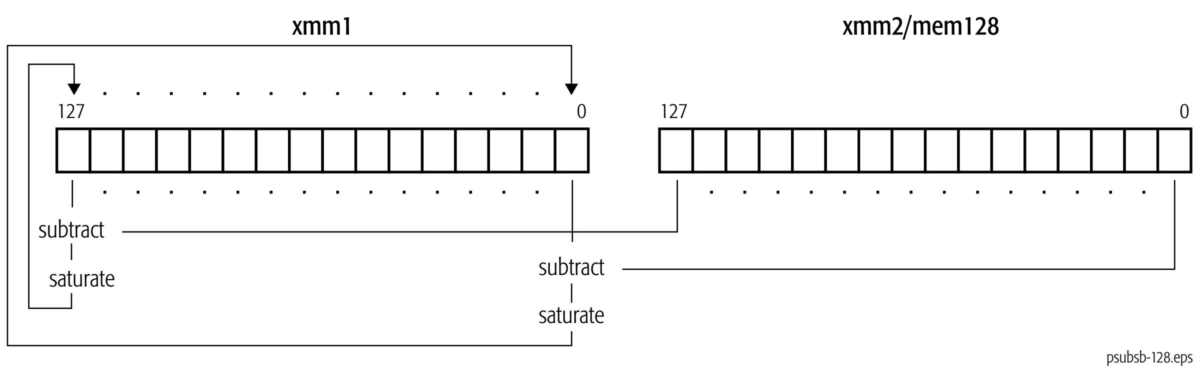
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBSB Packed Subtract Signed With Saturation Bytes

Subtracts each packed 8-bit signed integer value in the second source operand from the corresponding packed 8-bit signed integer in the first source operand and writes the signed integer result of each subtraction in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PSUBSB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBSB *xmm1*, *xmm2*/*mem128*	66 0F E8 /*r*	Subtracts packed byte signed integer values in an XMM register or 128-bit memory location from packed byte integer values in another XMM register and writes the result in the destination XMM register.



For each packed value in the destination, if the value is larger than the largest signed 8-bit integer, it is saturated to 7Fh, and if the value is smaller than the smallest signed 8-bit integer, it is saturated to 80h.

Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSW, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

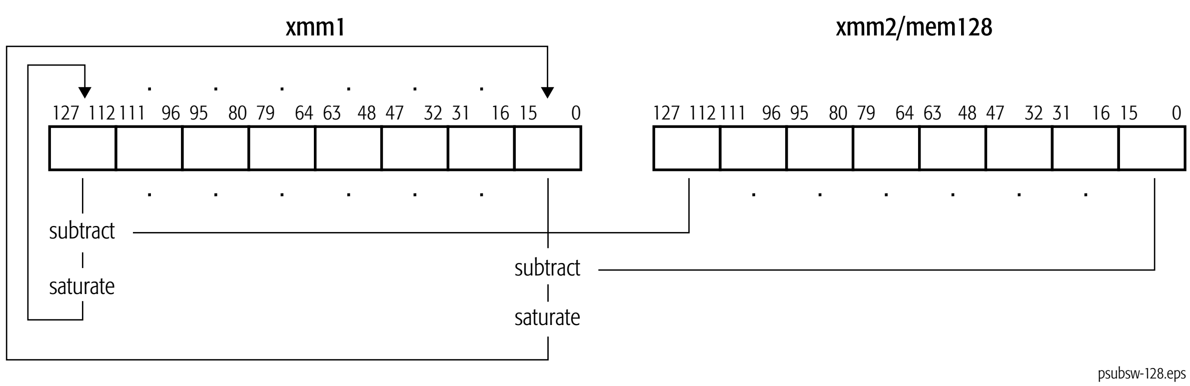
PSUBSW Packed Subtract Signed With Saturation Words

Subtracts each packed 16-bit signed integer value in the second source operand from the corresponding packed 16-bit signed integer in the first source operand and writes the signed integer result of each subtraction in the corresponding word of the destination (first source). The first source/destination and source operands are an XMM register and another XMM register or 128-bit memory location.

For each packed value in the destination, if the value is larger than the largest signed 16-bit integer, it is saturated to 7FFFh, and if the value is smaller than the smallest signed 16-bit integer, it is saturated to 8000h.

The PSUBSW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBSW *xmm1*, *xmm2/mem128*	66 0F E9 */r*	Subtracts packed 16-bit signed integer values in an XMM register or 128-bit memory location from packed 16-bit integer values in another XMM register and writes the result in the destination XMM register.



Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBUSB, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

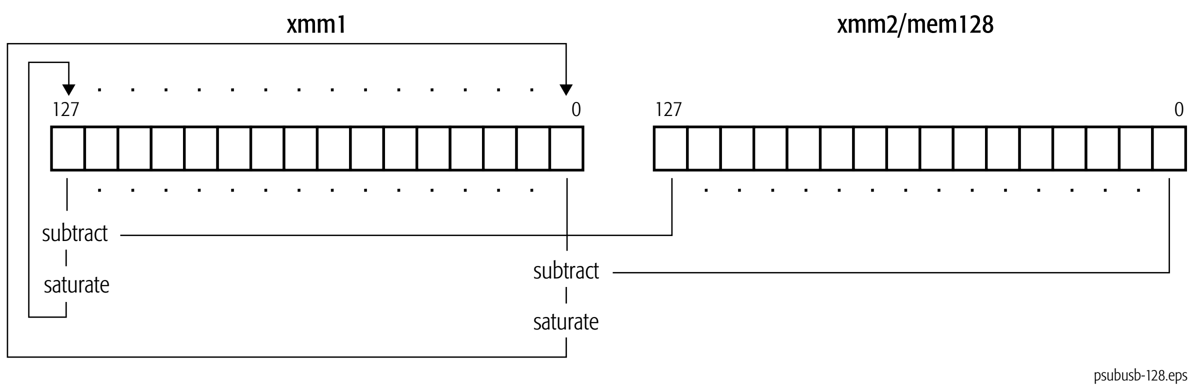
PSUBUSB Packed Subtract Unsigned and Saturate Bytes

Subtracts each packed 8-bit unsigned integer value in the second source operand from the corresponding packed 8-bit unsigned integer in the first source operand and writes the unsigned integer result of each subtraction in the corresponding byte of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

For each packed value in the destination, if the value is larger than the largest unsigned 8-bit integer, it is saturated to FFh, and if the value is smaller than the smallest unsigned 8-bit integer, it is saturated to 00h.

The PSUBUSB instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBUSB *xmm1*, *xmm2/mem128*	66 0F D8 /*r*	Subtracts packed byte unsigned integer values in an XMM register or 128-bit memory location from packed byte integer values in another XMM register and writes the result in the destination XMM register.



Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSW, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

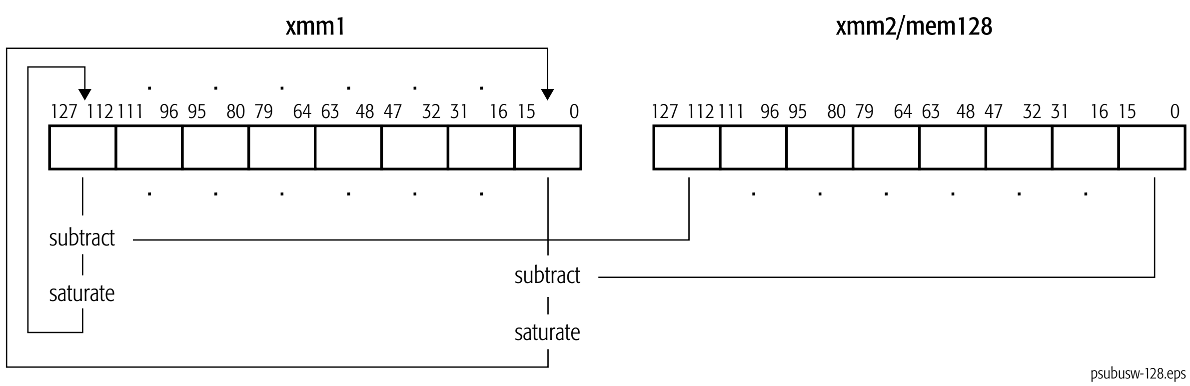
PSUBUSW Packed Subtract Unsigned and Saturate Words

Subtracts each packed 16-bit unsigned integer value in the second source operand from the corresponding packed 16-bit unsigned integer in the first source operand and writes the unsigned integer result of each subtraction in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

For each packed value in the destination, if the value is larger than the largest unsigned 16-bit integer, it is saturated to FFFFh, and if the value is smaller than the smallest unsigned 16-bit integer, it is saturated to 0000h.

The PSUBUSW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBUSW *xmm1*, *xmm2/mem128*	66 0F D9 /*r*	Subtracts packed 16-bit unsigned integer values in an XMM register or 128-bit memory location from packed 16-bit integer values in another XMM register and writes the result in the destination XMM register.



Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PSUBW Packed Subtract Words

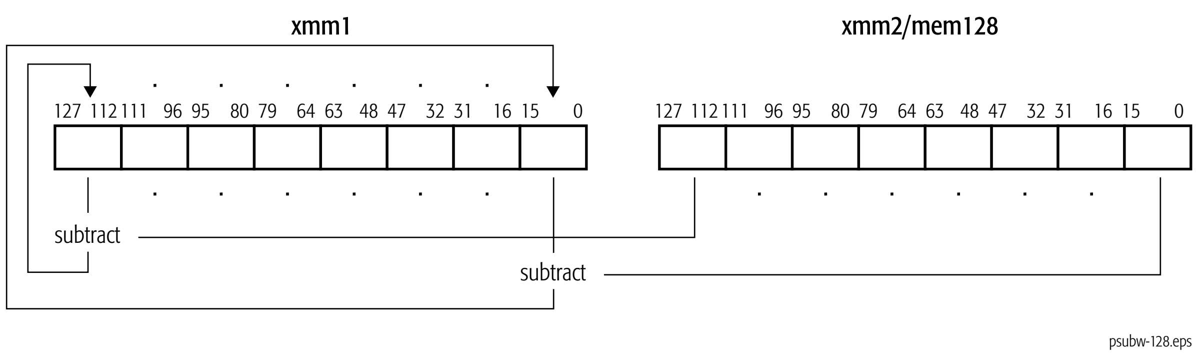
Subtracts each packed 16-bit integer value in the second source operand from the corresponding packed 16-bit integer in the first source operand and writes the integer result of each subtraction in the corresponding word of the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

For each packed value in the destination, if the value is larger than the largest unsigned 16-bit integer, it is saturated to FFFFh, and if the value is smaller than the smallest unsigned 16-bit integer, it is saturated to 0000h.

This instruction operates on both signed and unsigned integers. If the result overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set), and only the low-order 16 bits of the result are written in the destination.

The PSUBW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PSUBW *xmm1*, *xmm2/mem128*	66 0F F9 /*r*	Subtracts packed 16-bit integer values in an XMM register or 128-bit memory location from packed 16-bit integer values in another XMM register and writes the result in the destination XMM register.



Related Instructions

PSUBB, PSUBD, PSUBQ, PSUBSB, PSUBSW, PSUBUSB, PSUBUSW

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

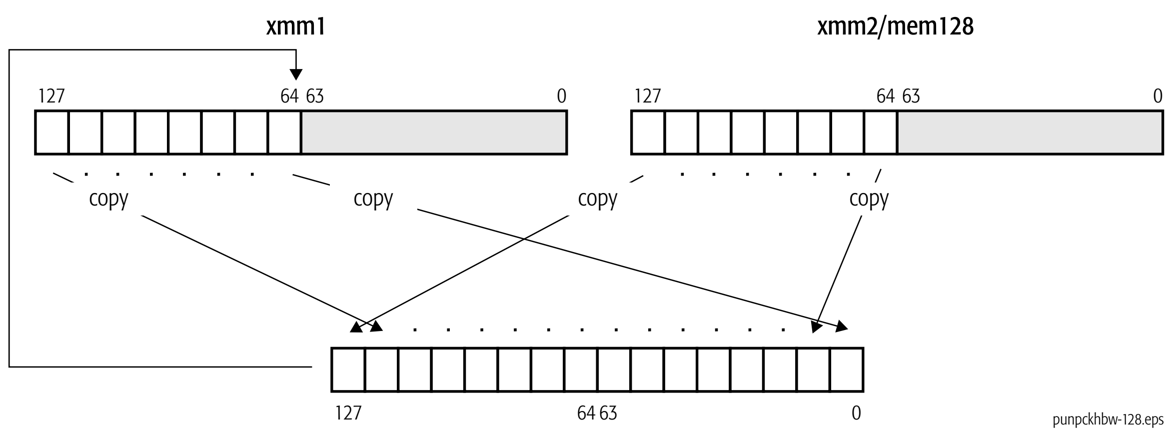
PUNPCKHBW Unpack and Interleave High Bytes

Unpacks the high-order bytes from the first and second source operands and packs them into interleaved bytes in the destination (first source). The low-order bytes of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

If the second source operand is all 0s, the destination contains the bytes from the first source operand zero-extended to 16 bits. This operation is useful for expanding unsigned 8-bit values to unsigned 16-bit operands for subsequent processing that requires higher precision.

The PUNPCKHBW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKHBW *xmm1*, *xmm2/mem128*	66 0F 68 /*r*	Unpacks the eight high-order bytes in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved bytes in the destination XMM register.



Related Instructions

PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

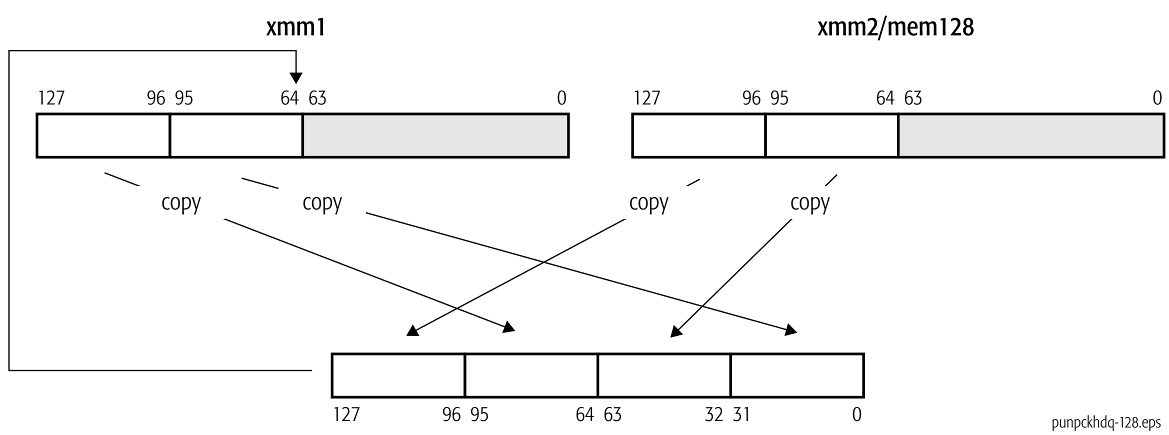
PUNPCKHDQ Unpack and Interleave High Doublewords

Unpacks the high-order doublewords from the first and second source operands and packs them into interleaved doublewords in the destination (first source). The low-order doublewords of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

If the second source operand is all 0s, the destination contains the doubleword(s) from the first source operand zero-extended to 64 bits. This operation is useful for expanding unsigned 32-bit values to unsigned 64-bit operands for subsequent processing that requires higher precision.

The PUNPCKHDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKHDQ *xmm1*, *xmm2/mem128*	66 0F 6A /*r*	Unpacks two high-order doublewords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved doublewords in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

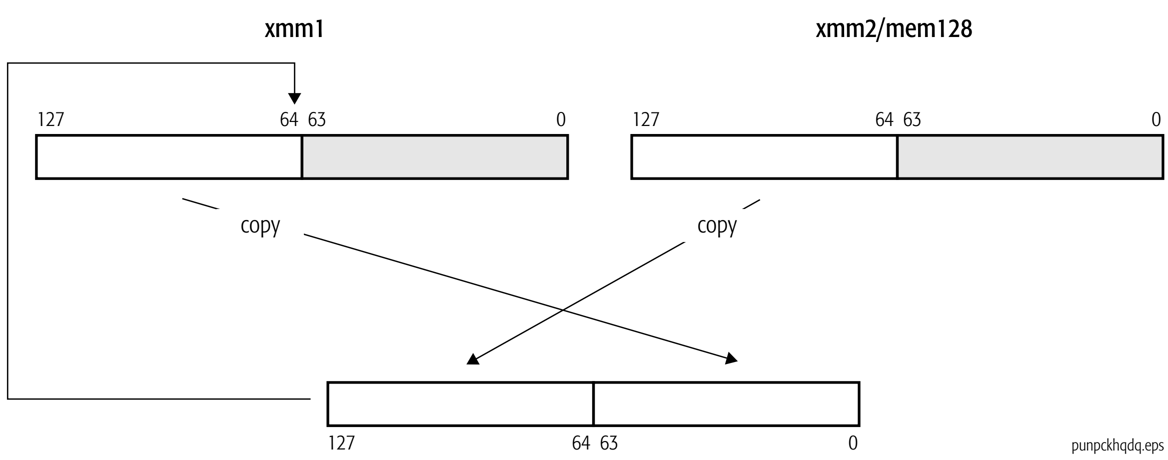
PUNPCKHQDQ Unpack and Interleave High Quadwords

Unpacks the high-order quadwords from the first and second source operands and packs them into interleaved quadwords in the destination (first source). The first source/destination is an XMM register, and the second source operand is another XMM register or 128-bit memory location. The low-order quadwords of the source operands are ignored.

If the second source operand is all 0s, the destination contains the quadword from the first source operand zero-extended to 128 bits. This operation is useful for expanding unsigned 64-bit values to unsigned 128-bit operands for subsequent processing that requires higher precision.

The PUNPCKHQDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKHQDQ *xmm1*, *xmm2/mem128*	66 0F 6D /*r*	Unpacks high-order quadwords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved quadwords in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

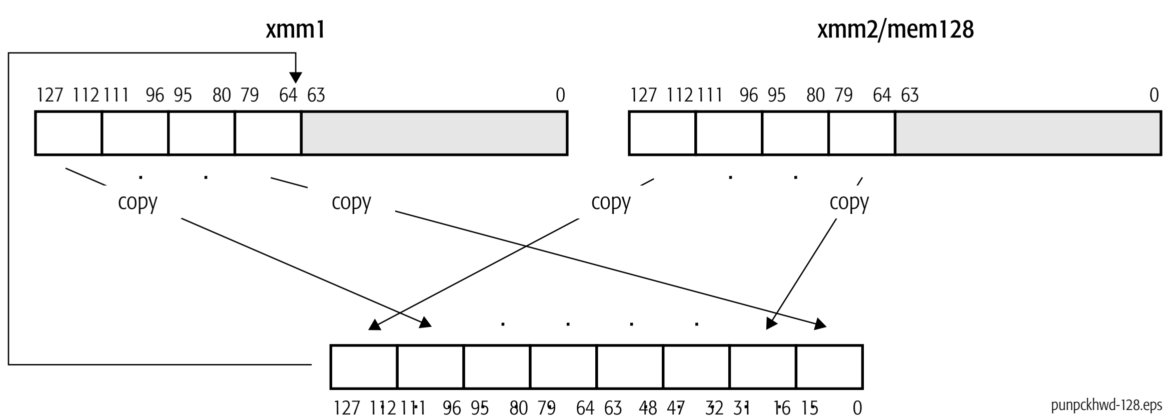
PUNPCKHWD — Unpack and Interleave High Words

Unpacks the high-order words from the first and second source operands and packs them into interleaved words in the destination (first source). The low-order words of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

If the second source operand is all 0s, the destination contains the words from the first source operand zero-extended to 32 bits. This operation is useful for expanding unsigned 16-bit values to unsigned 32-bit operands for subsequent processing that requires higher precision.

The PUNPCKHWD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKHWD *xmm1*, *xmm2/mem128*	66 0F 69 /r	Unpacks four high-order words in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved words in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

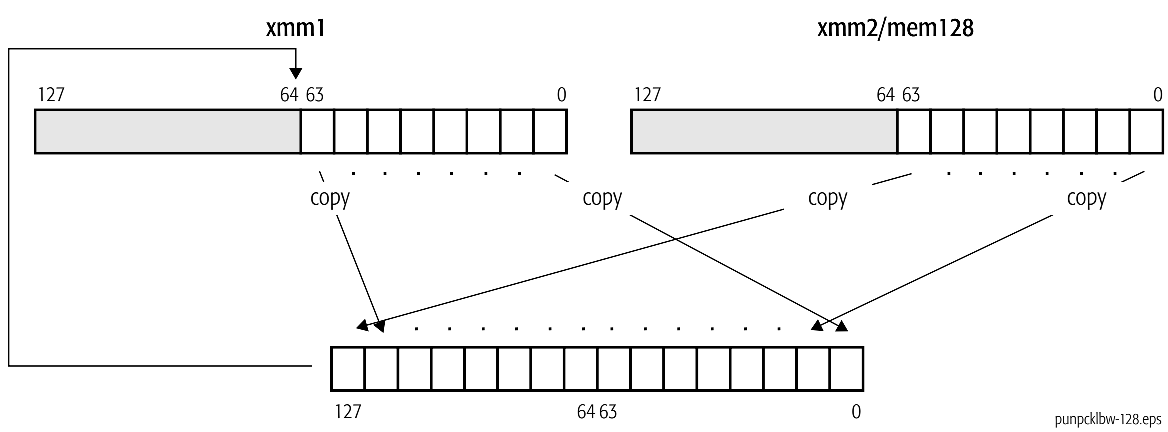
PUNPCKLBW Unpack and Interleave Low Bytes

Unpacks the low-order bytes from the first and second source operands and packs them into interleaved bytes in the destination (first source). The high-order bytes of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

If the second source operand is all 0s, the destination contains the bytes from the first source operand zero-extended to 16 bits. This operation is useful for expanding unsigned 8-bit values to unsigned 16-bit operands for subsequent processing that requires higher precision.

The PUNPCKLBW instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKLBW *xmm1*, *xmm2/mem128*	66 0F 60 /*r*	Unpacks the eight low-order bytes in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved bytes in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLDQ, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

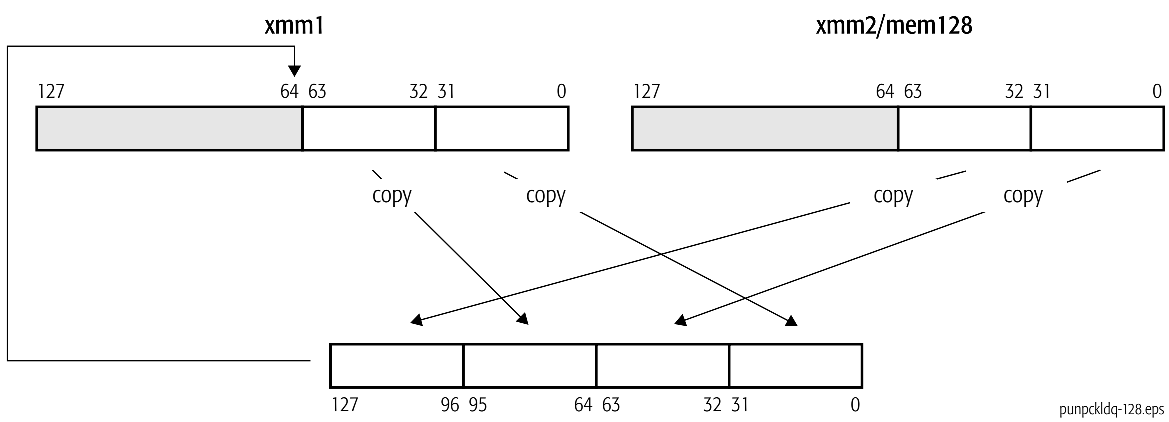
PUNPCKLDQ Unpack and Interleave Low Doublewords

Unpacks the low-order doublewords from the first and second source operands and packs them into interleaved doublewords in the destination (first source). The high-order doublewords of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

If the second source operand is all 0s, the destination contains the doubleword(s) from the first source operand zero-extended to 64 bits. This operation is useful for expanding unsigned 32-bit values to unsigned 64-bit operands for subsequent processing that requires higher precision.

The PUNPCKLDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKLDQ *xmm1, xmm2/mem128*	66 0F 62 */r*	Unpacks two low-order doublewords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved doublewords in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLQDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

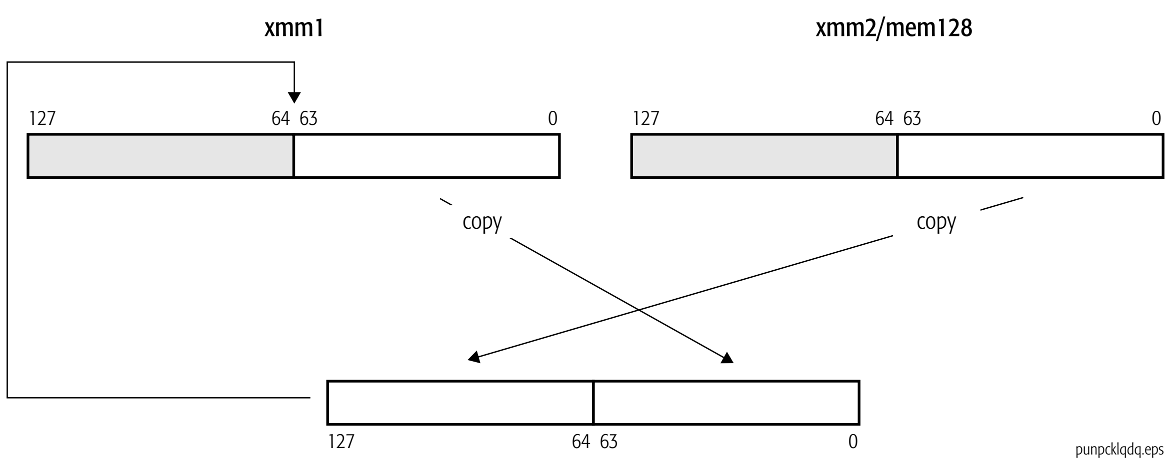
PUNPCKLQDQ Unpack and Interleave Low Quadwords

Unpacks the low-order quadwords from the first and second source operands and packs them into interleaved quadwords in the destination (first source). The first source/destination is an XMM register, and the second source operand is another XMM register or 128-bit memory location. The high-order quadwords of the source operands are ignored.

If the second source operand is all 0s, the destination contains the quadword from the first source operand zero-extended to 128 bits. This operation is useful for expanding unsigned 64-bit values to unsigned 128-bit operands for subsequent processing that requires higher precision.

The PUNPCKLQDQ instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKLQDQ *xmm1*, *xmm2/mem128*	66 0F 6C /*r*	Unpacks low-order quadwords in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved quadwords in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

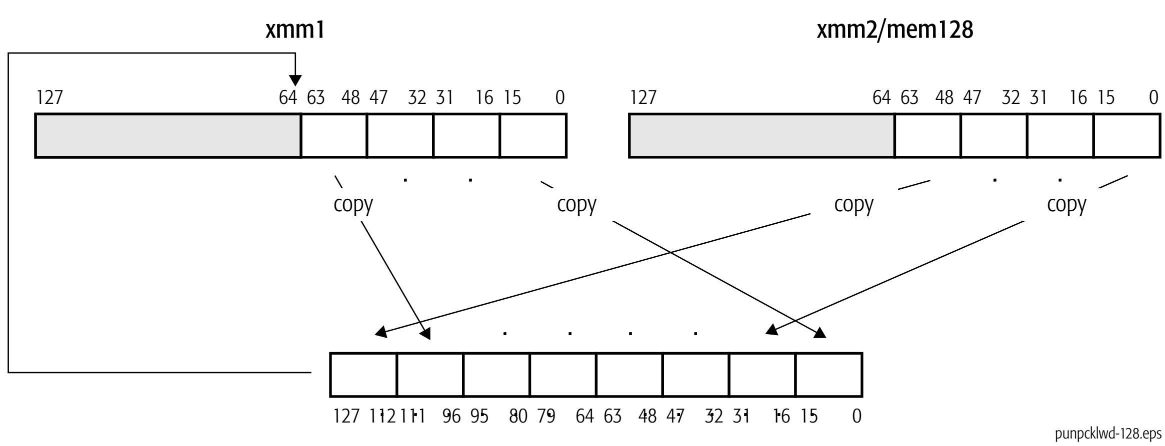
PUNPCKLWD Unpack and Interleave Low Words

Unpacks the low-order words from the first and second source operands and packs them into interleaved words in the destination (first source). The high-order words of the source operands are ignored. The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

If the second source operand is all 0s, the destination contains the words from the first source operand zero-extended to 32 bits. This operation is useful for expanding unsigned 16-bit values to unsigned 32-bit operands for subsequent processing that requires higher precision.

The PUNPCKLWD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
PUNPCKLWD *xmm1*, *xmm2/mem128*	66 0F 61 */r*	Unpacks the four low-order words in an XMM register and another XMM register or 128-bit memory location and packs them into interleaved words in the destination XMM register.



Related Instructions

PUNPCKHBW, PUNPCKHDQ, PUNPCKHQDQ, PUNPCKHWD, PUNPCKLBW, PUNPCKLDQ, PUNPCKLQDQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

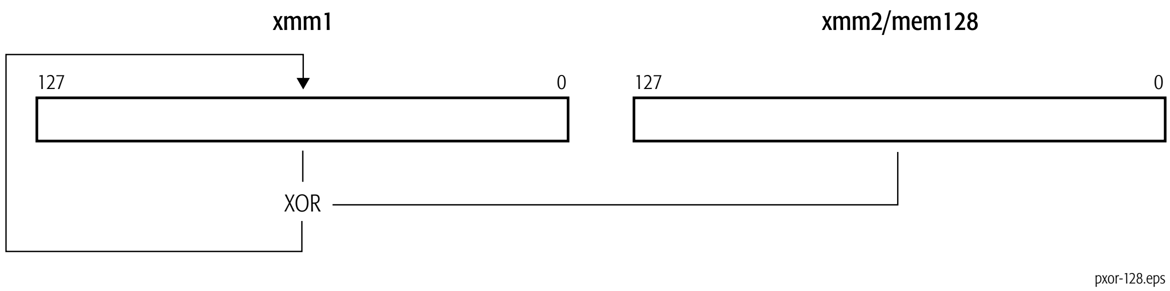
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

PXOR Packed Logical Bitwise Exclusive OR

Performs a bitwise exclusive OR of the values in the first and second source operands and writes the result in the destination (first source). The first source/destination operand is an XMM register and the second source operand is another XMM register or 128-bit memory location.

The PXOR instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
PXOR *xmm1*, *xmm2/mem128*	66 0F EF /*r*	Performs bitwise logical XOR of values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

PAND, PANDN, POR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

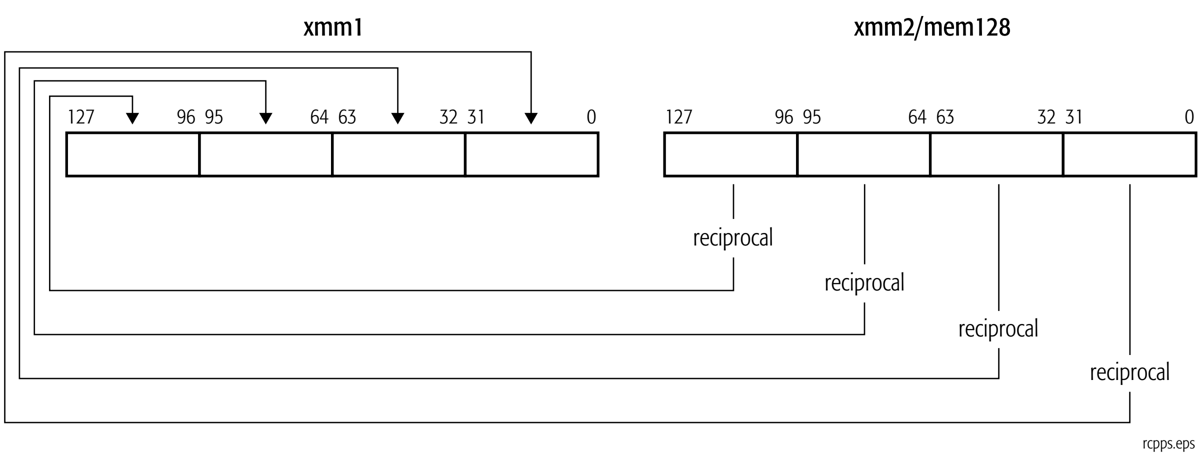
RCPPS Reciprocal Packed Single-Precision Floating-Point

Computes the approximate reciprocal of each of the four packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding doubleword of another XMM register. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum error is less than or equal to $1.5 * 2^{-12}$ times the true reciprocal. A source value that is ±zero or denormal returns an infinity of the source value's sign. Results that underflow are changed to signed zero. For both SNaN and QNaN source operands, a QNaN is returned.

The RCPPS instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
RCPPS *xmm1*, *xmm2/mem128*	0F 53 /*r*	Computes reciprocals of packed single-precision floating-point values in an XMM register or 128-bit memory location and writes result in the destination XMM register.



Related Instructions

RCPSS, RSQRTPS, RSQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

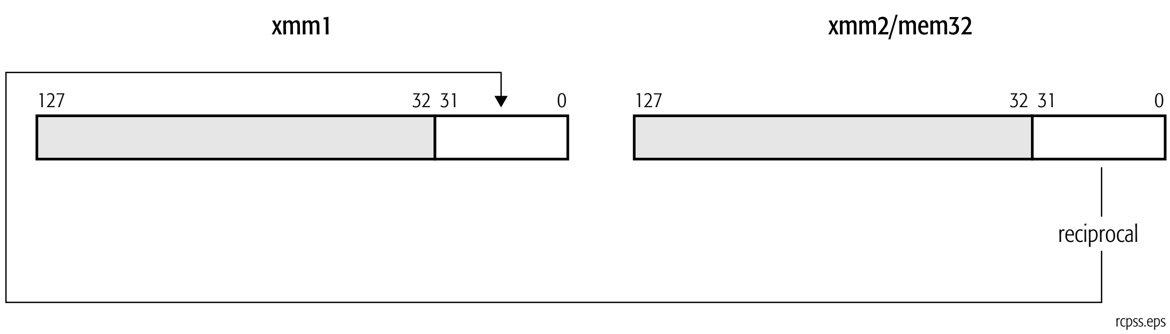
RCPSS Reciprocal Scalar Single-Precision Floating-Point

Computes the approximate reciprocal of the low-order single-precision floating-point value in an XMM register or in a 32-bit memory location and writes the result in the low-order doubleword of another XMM register. The three high-order doublewords in the destination XMM register are not modified. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum error is less than or equal to $1.5 * 2^{-12}$ times the true reciprocal. A source value that is ±zero or denormal returns an infinity of the source value's sign. Results that underflow are changed to signed zero. For both SNaN and QNaN source operands, a QNaN is returned.

The RCPSS instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
RCPSS *xmm1*, *xmm2/mem32*	F3 0F 53 */r*	Computes reciprocal of scalar single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

RCPPS, RSQRTPS, RSQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

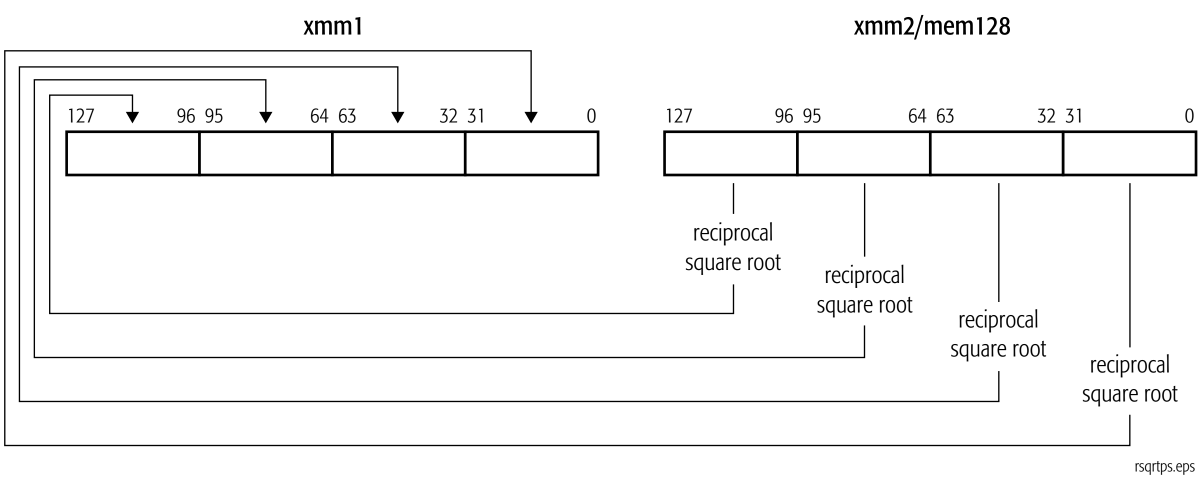
RSQRTPS Reciprocal Square Root Packed Single-Precision Floating-Point

Computes the approximate reciprocal of the square root of each of the four packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding doubleword of another XMM register. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum error is less than or equal to $1.5 * 2^{-12}$ times the true reciprocal square root. A source value that is ±zero or denormal returns an infinity of the source value's sign. Negative source values other than –zero and –denormal return a QNaN floating-point indefinite value ("Indefinite Values" in Volume 1). For both SNaN and QNaN source operands, a QNaN is returned.

The RSQRTPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
RSQRTPS *xmm1*, *xmm2/mem128*	0F 52 /*r*	Computes reciprocals of square roots of packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTSS, SQRTPD, SQRTPS, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

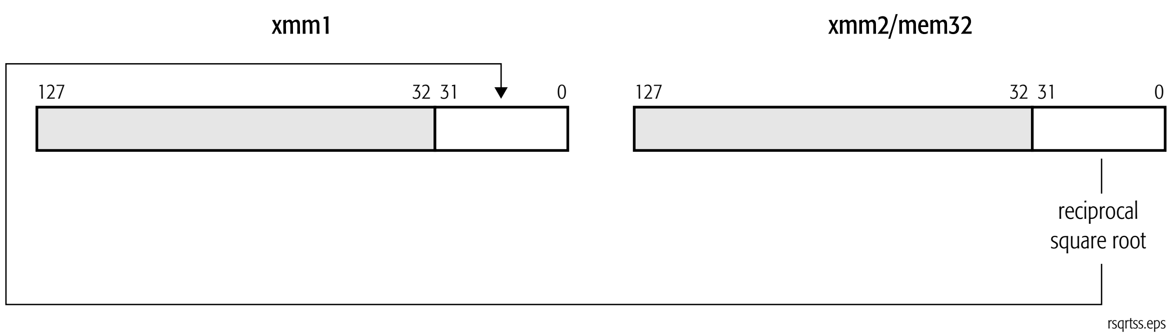
RSQRTSS Reciprocal Square Root Scalar Single-Precision Floating-Point

Computes the approximate reciprocal of the square root of the low-order single-precision floating-point value in an XMM register or in a 32-bit memory location and writes the result in the low-order doubleword of another XMM register. The three high-order doublewords in the destination XMM register are not modified. The rounding control bits (RC) in the MXCSR register have no effect on the result.

The maximum error is less than or equal to $1.5 * 2^{-12}$ times the true reciprocal square root. A source value that is ±zero or denormal returns an infinity of the source value's sign. Negative source values other than –zero and –denormal return a QNaN floating-point indefinite value ("Indefinite Values" in Volume 1). For both SNaN and QNaN source operands, a QNaN is returned.

The RSQRTSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
RSQRTSS *xmm1*, *xmm2/mem32*	F3 0F 52 */r*	Computes reciprocal of square root of single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, SQRTPD, SQRTPS, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

SHUFPD Shuffle Packed Double-Precision Floating-Point

Moves either of the two packed double-precision floating-point values in the first source operand to the low-order quadword of the destination (first source) and moves either of the two packed double-precision floating-point values in the second source operand to the high-order quadword of the destination. In each case, the value of the destination quadword is determined by the least-significant two bits in the immediate-byte operand, as shown in Table 1-7 on page 380. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The SHUFPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
SHUFPD *xmm1*, *xmm2/mem128*, *imm8*	66 0F C6 */r ib*	Shuffles packed double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and puts the result in the destination XMM register.

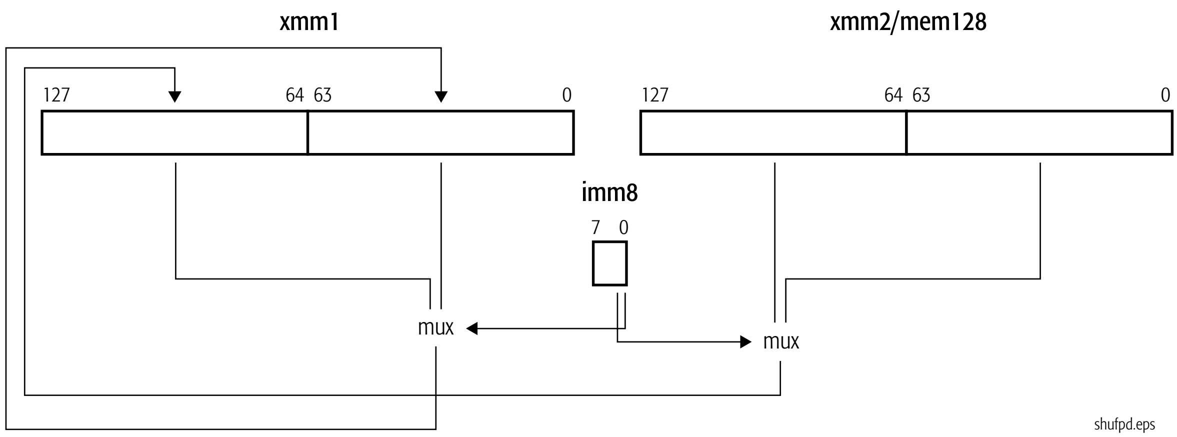


Table 1-7. Immediate-Byte Operand Encoding for SHUFPD

Destination Bits Filled	Immediate-Byte Bit Field	Value of Bit Field	Source 1 Bits Moved	Source 2 Bits Moved
63–0	0	0	63–0	–
		1	127–64	–
127–64	1	0	–	63–0
		1	–	127–64

Related Instructions

SHUFPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

SHUFPS Shuffle Packed Single-Precision Floating-Point

Moves two of the four packed single-precision floating-point values in the first source operand to the low-order quadword of the destination (first source) and moves two of the four packed single-precision floating-point values in the second source operand to the high-order quadword of the destination. In each case, the value of the destination doubleword is determined by a two-bit field in the immediate-byte operand, as shown in Table 1-8 on page 382. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The SHUFPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
SHUFPS *xmm1*, *xmm2/mem128*, *imm8*	0F C6 */r ib*	Shuffles packed single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and puts the result in the destination XMM register.

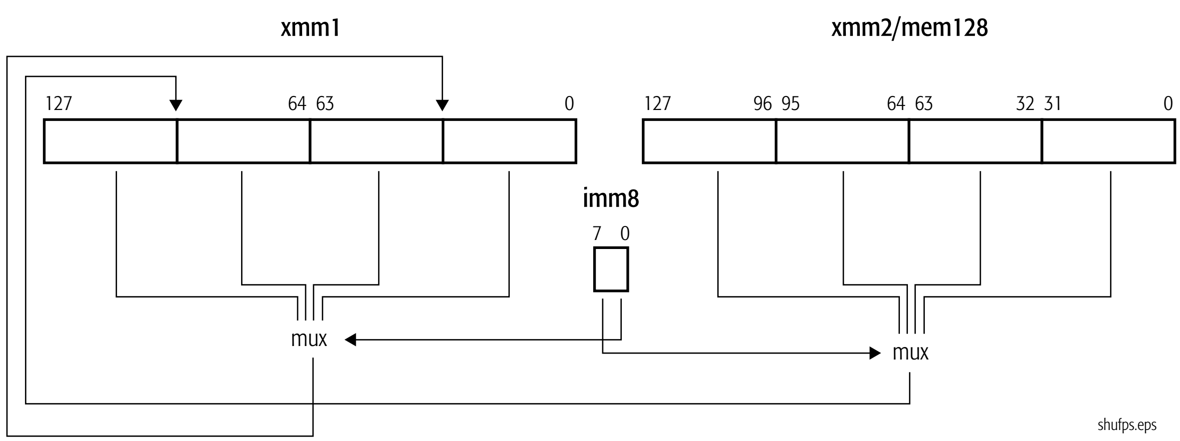


Table 1-8. Immediate-Byte Operand Encoding for SHUFPS

Destination Bits Filled	Immediate-Byte Bit Field	Value of Bit Field	Source 1 Bits Moved	Source 2 Bits Moved
31–0	1–0	0	31–0	–
		1	63–32	–
		2	95–64	–
		3	127–96	–
63–32	3–2	0	31–0	–
		1	63–32	–
		2	95–64	–
		3	127–96	–
95–64	5–4	0	–	31–0
		1	–	63–32
		2	–	95–64
		3	–	127–96
127–96	7–6	0	–	31–0
		1	–	63–32
		2	–	95–64
		3	–	127–96

Related Instructions

SHUFPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

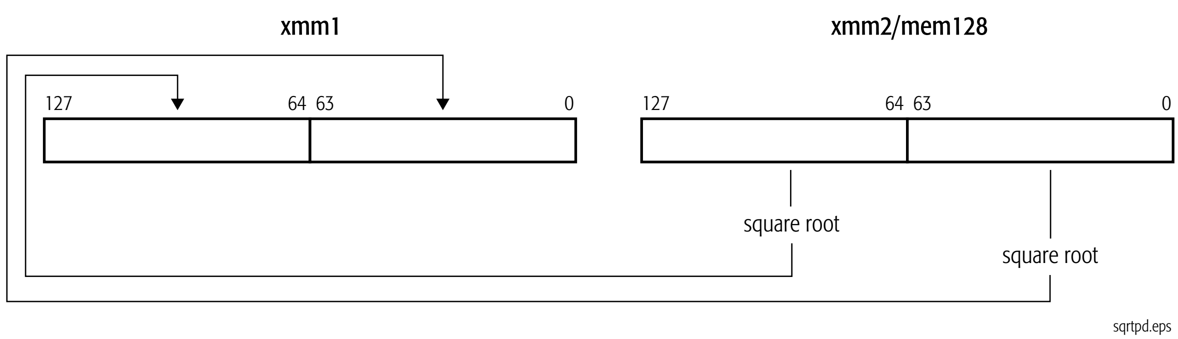
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

SQRTPD Square Root Packed Double-Precision Floating-Point

Computes the square root of each of the two packed double-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding quadword of another XMM register. Taking the square root of +infinity returns +infinity.

The SQRTPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
SQRTPD *xmm1*, *xmm2/mem128*	66 0F 51 */r*	Computes square roots of packed double-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, RSQRTSS, SQRTPS, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M				M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

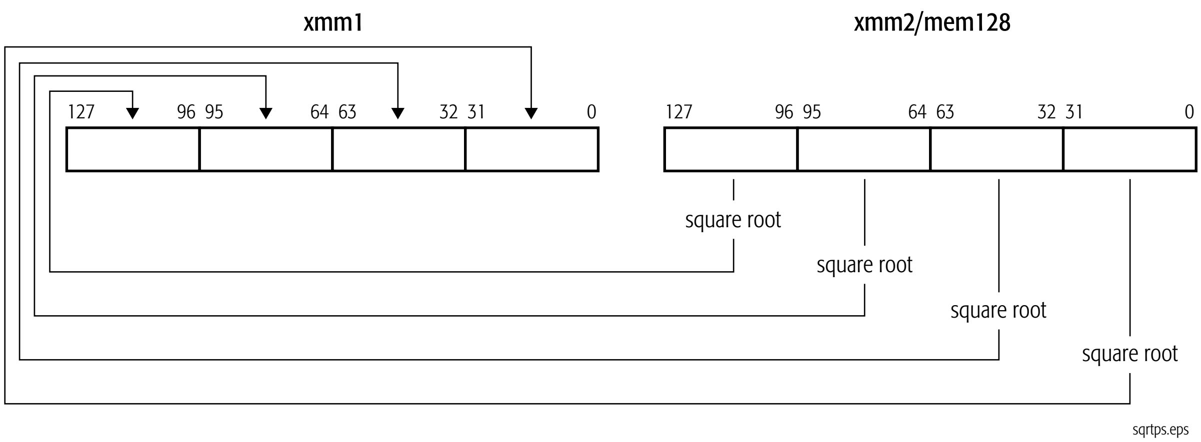
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was negative (not including –0).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

SQRTPS Square Root Packed Single-Precision Floating-Point

Computes the square root of each of the four packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the corresponding doubleword of another XMM register. Taking the square root of +infinity returns +infinity.

The SQRTPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
SQRTPS *xmm1*, *xmm2/mem128*	0F 51 */r*	Computes square roots of packed single-precision floating-point values in an XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, RSQRTSS, SQRTPD, SQRTSD, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M				M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

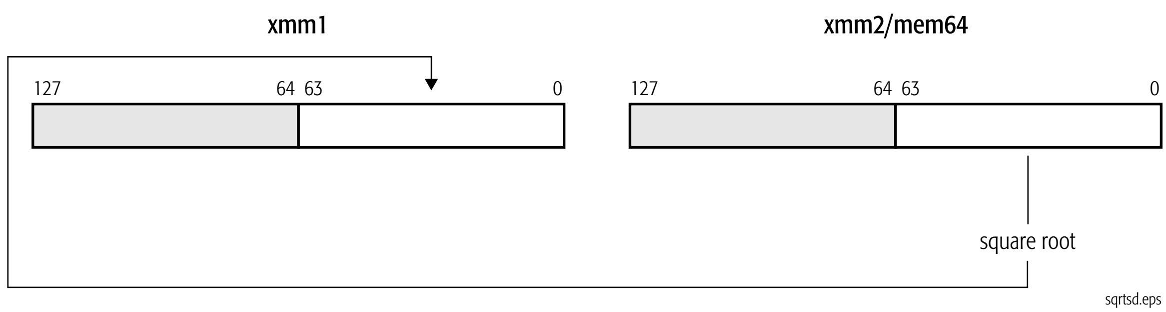
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was negative (not including –0).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

SQRTSD Square Root Scalar Double-Precision Floating-Point

Computes the square root of the low-order double-precision floating-point value in an XMM register or in a 64-bit memory location and writes the result in the low-order quadword of another XMM register. The high-order quadword of the destination XMM register is not modified. Taking the square root of +infinity returns +infinity.

The SQRTSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
SQRTSD *xmm1*, *xmm2/mem64*	F2 0F 51 /*r*	Computes square root of double-precision floating-point value in an XMM register or 64-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, RSQRTSS, SQRTPD, SQRTPS, SQRTSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M				M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions.

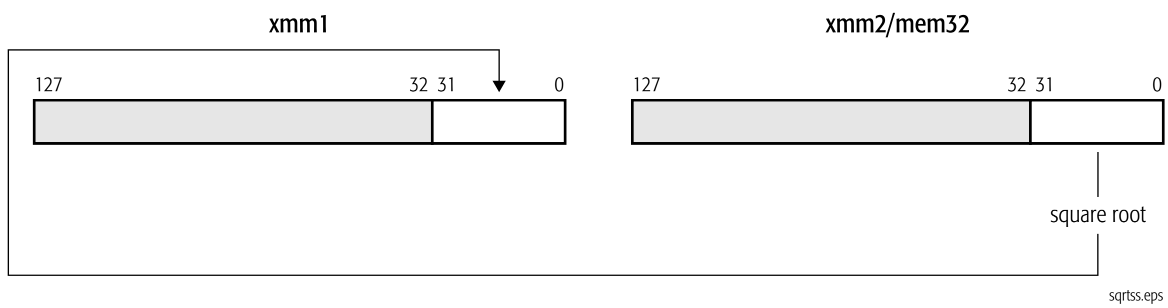
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was negative (not including –0).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

SQRTSS Square Root Scalar Single-Precision Floating-Point

Computes the square root of the low-order single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the low-order doubleword of another XMM register. The three high-order doublewords of the destination XMM register are not modified. Taking the square root of +infinity returns +infinity.

The SQRTSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
SQRTSS *xmm1*, *xmm2/mem32*	F3 0F 51 */r*	Computes square root of single-precision floating-point value in an XMM register or 32-bit memory location and writes the result in the destination XMM register.



Related Instructions

RSQRTPS, RSQRTSS, SQRTPD, SQRTPS, SQRTSD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M				M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	A source operand was negative (not including –0).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

STMXCSR Store MXCSR Control/Status Register

Saves the contents of the MXCSR register in a 32-bit location in memory. The MXCSR register is described in “Registers” in Volume 1.

The STMXCSR instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
STMXCSR *mem32*	0F AE /3	Stores contents of MXCSR in 32-bit memory location.

Related Instructions

LDMXCSR

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

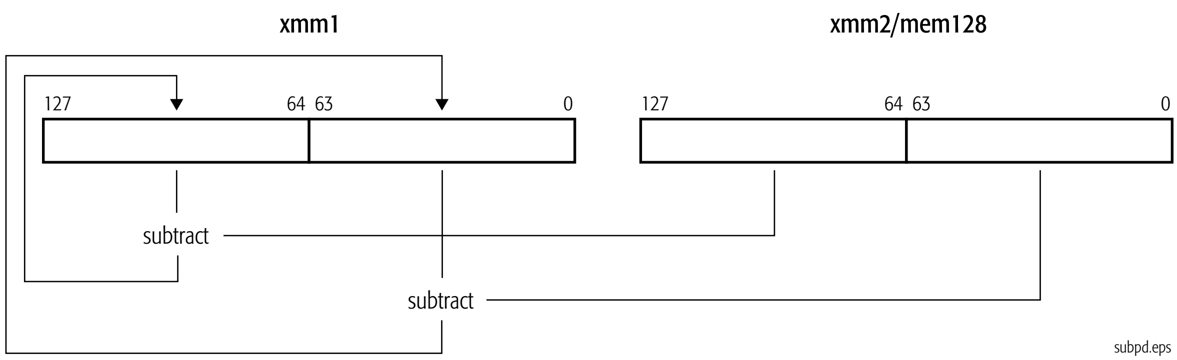
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
			X	The destination operand was in a non-writable segment.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.

SUBPD Subtract Packed Double-Precision Floating-Point

Subtracts each packed double-precision floating-point value in the second source operand from the corresponding packed double-precision floating-point value in the first source operand and writes the result of each subtraction in the corresponding quadword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The SUBPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
SUBPD *xmm1*, *xmm2/mem128*	66 0F 5C /*r*	Subtracts packed double-precision floating-point values in an XMM register or 128-bit memory location from packed double-precision floating-point values in another XMM register and writes the result in the destination XMM register.



Related Instructions

SUBPS, SUBSD, SUBSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

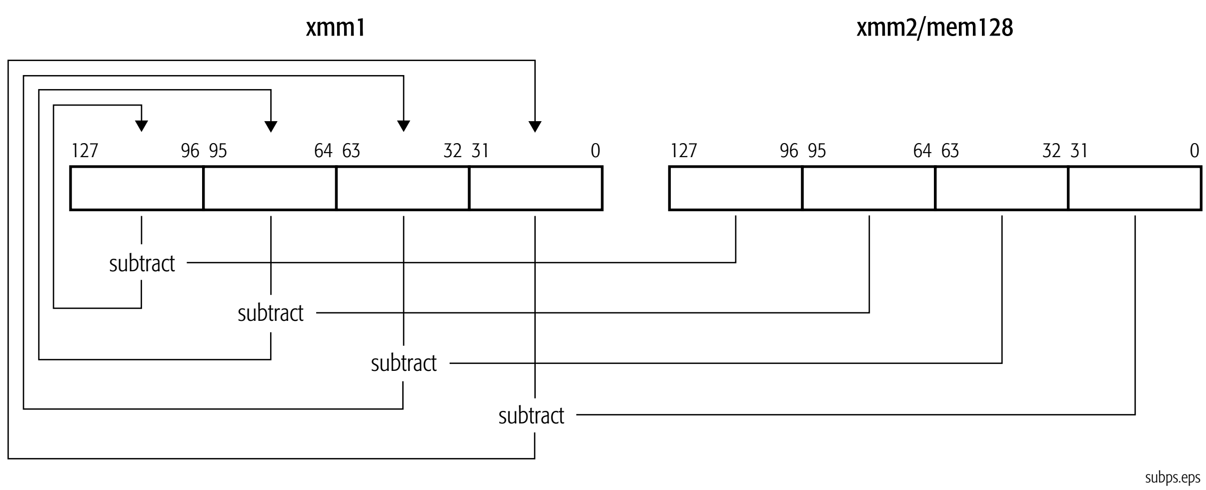
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

SUBPS Subtract Packed Single-Precision Floating-Point

Subtracts each packed single-precision floating-point value in the second source operand from the corresponding packed single-precision floating-point value in the first source operand and writes the result of each subtraction in the corresponding doubleword of the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The SUBPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
SUBPS *xmm1*, *xmm2/mem128*	0F 5C /*r*	Subtracts packed single-precision floating-point values in an XMM register or 128-bit memory location from packed single-precision floating-point values in another XMM register and writes the result in the destination XMM register.



Related Instructions

SUBPD, SUBSD, SUBSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

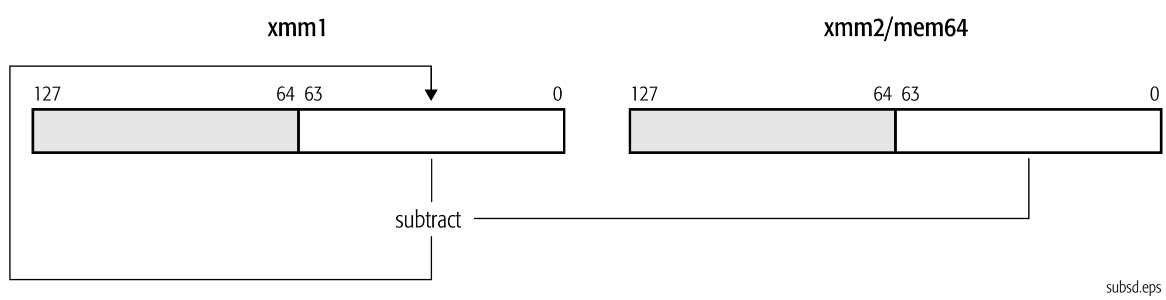
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

SUBSD Subtract Scalar Double-Precision Floating-Point

Subtracts the double-precision floating-point value in the low-order quadword of the second source operand from the double-precision floating-point value in the low-order quadword of the first source operand and writes the result in the low-order quadword of the destination (first source). The high-order quadword of the destination is not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 64-bit memory location.

The SUBSD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
SUBSD *xmm1*, *xmm2/mem64*	F2 0F 5C /*r*	Subtracts low-order double-precision floating-point value in an XMM register or in a 64-bit memory location from low-order double-precision floating-point value in another XMM register and writes the result in the destination XMM register.



Related Instructions

SUBPD, SUBPS, SUBSS

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:

A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

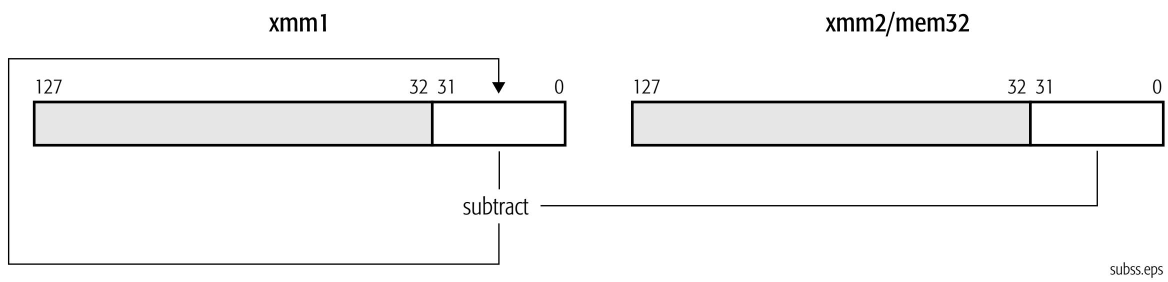
Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

SUBSS Subtract Scalar Single-Precision Floating-Point

Subtracts the single-precision floating-point value in the low-order doubleword of the second source operand from the single-precision floating-point value in the low-order doubleword of the first source operand and writes the result in the low-order doubleword of the destination (first source). The three high-order doublewords of the destination are not modified. The first source/destination operand is an XMM register. The second source operand is another XMM register or 32-bit memory location.

The SUBSS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
SUBSS *xmm1*, *xmm2/mem32*	F3 0F 5C /*r*	Subtracts low-order single-precision floating-point value in an XMM register or in a 32-bit memory location from low-order single-precision floating-point value in another XMM register and writes the result in the destination XMM register.



Related Instructions

SUBPD, SUBPS, SUBSD

rFLAGS Affected

None

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
										M	M	M		M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+infinity was subtracted from +infinity.
	X	X	X	–infinity was subtracted from –infinity.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

Exception	Real	Virtual 8086	Protected	Cause of Exception
Overflow exception (OE)	X	X	X	A rounded result was too large to fit into the format of the destination operand.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.
Precision exception (PE)	X	X	X	A result could not be represented exactly in the destination format.

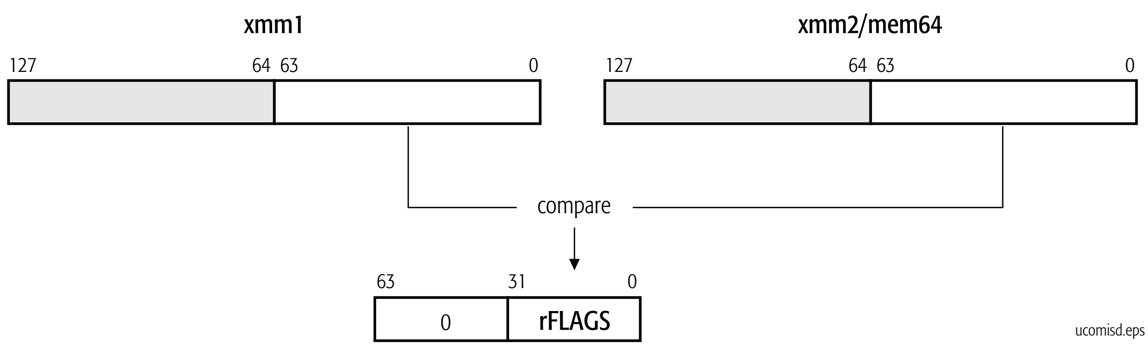
UCOMISD — Unordered Compare Scalar Double-Precision Floating-Point

Performs an unordered compare of the double-precision floating-point value in the low-order 64 bits of an XMM register with the double-precision floating-point value in the low-order 64 bits of another XMM register or a 64-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result of the compare. The OF, AF, and SF bits in rFLAGS are set to zero. The result is unordered if one or both of the operand values is a NaN.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

The UCOMISD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
UCOMISD *xmm1*, *xmm2/mem64*	66 0F 2E */r*	Compares scalar double-precision floating-point values in an XMM register and an XMM register or 64-bit memory location. Sets rFLAGS.



Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISD, COMISS, UCOMISS

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:
Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

UCOMISS

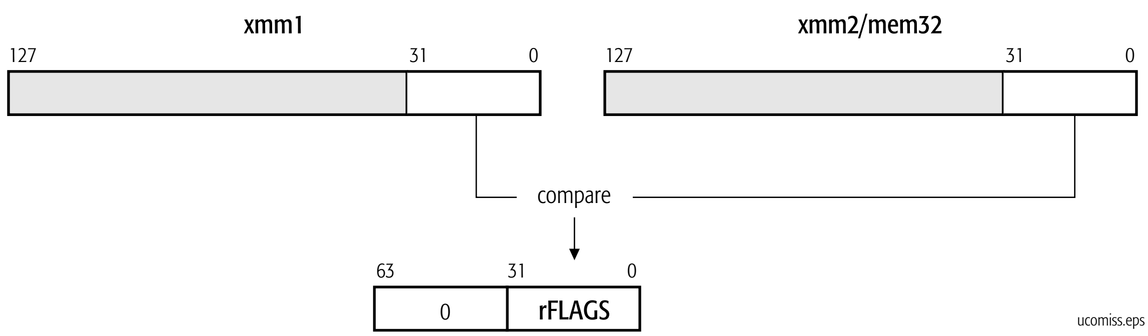
Unordered Compare Scalar Single-Precision Floating-Point

Performs an unordered compare of the single-precision floating-point value in the low-order 32 bits of an XMM register with the single-precision floating-point value in the low-order 32 bits of another XMM register or a 32-bit memory location and sets the ZF, PF, and CF bits in the rFLAGS register to reflect the result. The OF, AF, and SF bits in rFLAGS are set to zero. The result is unordered if one or both of the operand values is a NaN.

If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

The UCOMISS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
UCOMISS *xmm1*, *xmm2/mem32*	0F 2E /*r*	Compares scalar single-precision floating-point values in an XMM register and an XMM register or 32-bit memory location. Sets rFLAGS.



Result of Compare	ZF	PF	CF
Unordered	1	1	1
Greater Than	0	0	0
Less Than	0	0	1
Equal	1	0	0

Related Instructions

CMPPD, CMPPS, CMPSD, CMPSS, COMISD, COMISS, UCOMISD

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	M	M
21	20	19	18	17	16	14	13–12	11	10	9	8	7	6	4	2	0

Note:
Bits 31–22, 15, 5, 3, and 1 are reserved. A flag set to 1 or cleared to 0 is M (modified). Unaffected flags are blank.
If the instruction causes an unmasked SIMD floating-point exception (#XF), the rFLAGS bits are not updated.

MXCSR Flags Affected

FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
														M	M
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Note:
A flag that may be set to one or cleared to zero is M (modified). Unaffected flags are blank.

Exceptions

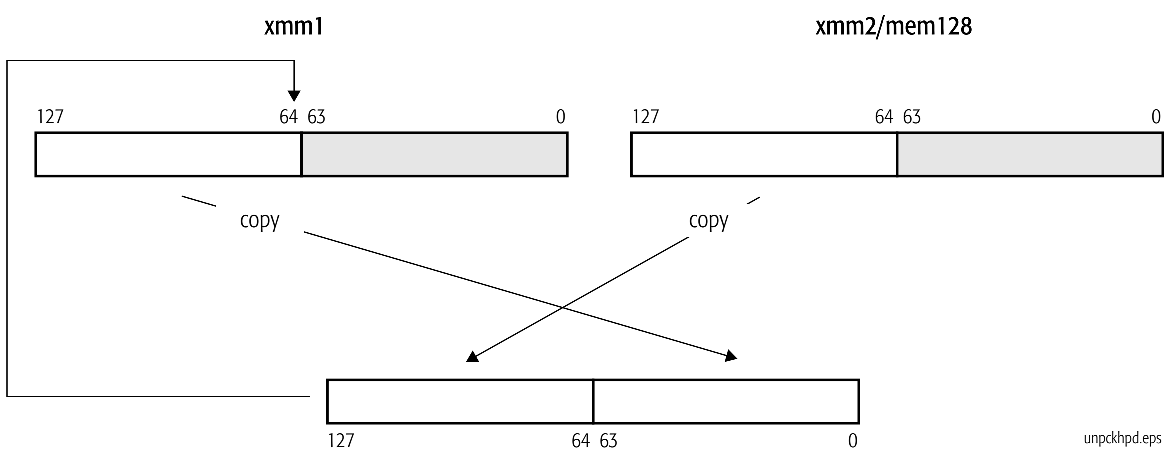
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 0. See *SIMD Floating-Point Exceptions*, below, for details.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled.
SIMD Floating-Point Exception, #XF	X	X	X	There was an unmasked SIMD floating-point exception while CR4.OSXMMEXCPT = 1. See *SIMD Floating-Point Exceptions*, below, for details.
SIMD Floating-Point Exceptions				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

UNPCKHPD Unpack High Double-Precision Floating-Point

Unpacks the high-order double-precision floating-point values in the first and second source operands and packs them into quadwords in the destination (first source). The value from the first source operand is packed into the low-order quadword of the destination, and the value from the second source operand is packed into the high-order quadword of the destination. The low-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The UNPCKHPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
UNPCKHPD *xmm1*, *xmm2/mem128*	66 0F 15 /*r*	Unpacks high-order double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



Related Instructions

UNPCKHPS, UNPCKLPD, UNPCKLPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

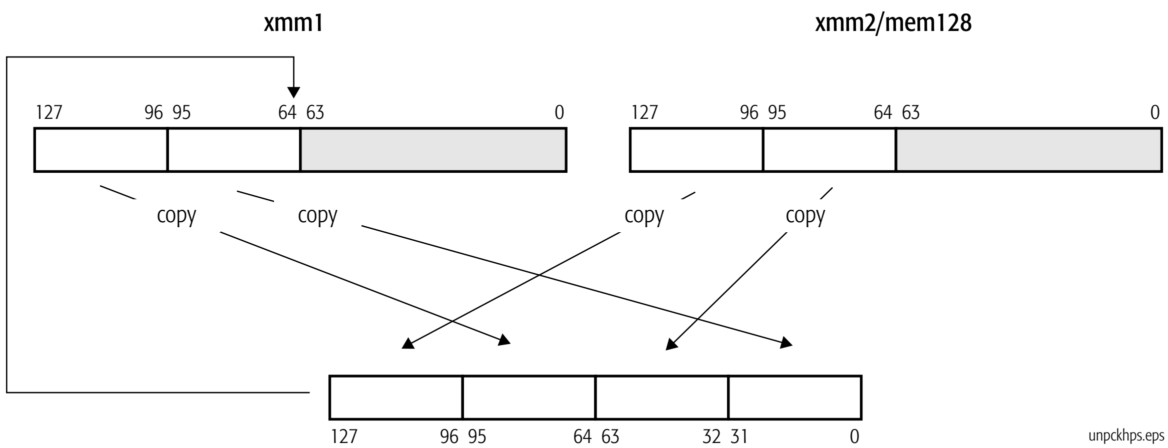
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDXX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

UNPCKHPS Unpack High Single-Precision Floating-Point

Unpacks the high-order single-precision floating-point values in the first and second source operands and packs them into interleaved doublewords in the destination (first source). The low-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The UNPCKHPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
UNPCKHPS *xmm1*, *xmm2/mem128*	0F 15 */r*	Unpacks high-order single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



Related Instructions

UNPCKHPD, UNPCKLPD, UNPCKLPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

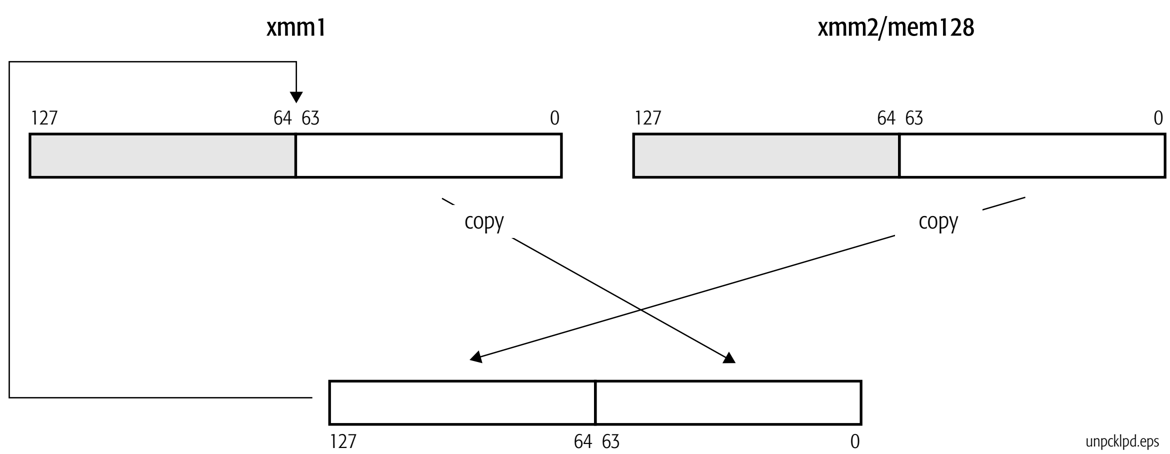
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

UNPCKLPD Unpack Low Double-Precision Floating-Point

Unpacks the low-order double-precision floating-point values in the first and second source operands and packs them into the destination (first source). The value from the first source operand is packed into the low-order quadword of the destination, and the value from the second source operand is packed into the high-order quadword of the destination. The high-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The UNPCKLPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See "CPUID" in Volume 3.)

Mnemonic	Opcode	Description
UNPCKLPD *xmm1*, *xmm2/mem128*	66 0F 14 /*r*	Unpacks low-order double-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



Related Instructions

UNPCKHPD, UNPCKHPS, UNPCKLPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

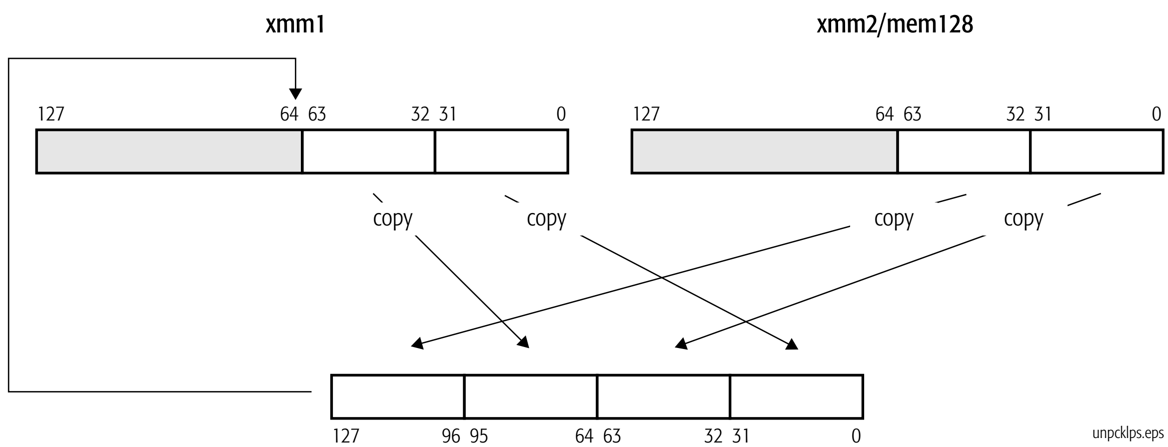
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

UNPCKLPS Unpack Low Single-Precision Floating-Point

Unpacks the low-order single-precision floating-point values in the first and second source operands and packs them into interleaved doublewords in the destination (first source). The high-order quadwords of the source operands are ignored. The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location

The UNPCKLPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
UNPCKLPS *xmm1*, *xmm2/mem128*	0F 14 /*r*	Unpacks low-order single-precision floating-point values in an XMM register and another XMM register or 128-bit memory location and packs them into the destination XMM register.



Related Instructions

UNPCKHPD, UNPCKHPS, UNPCKLPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

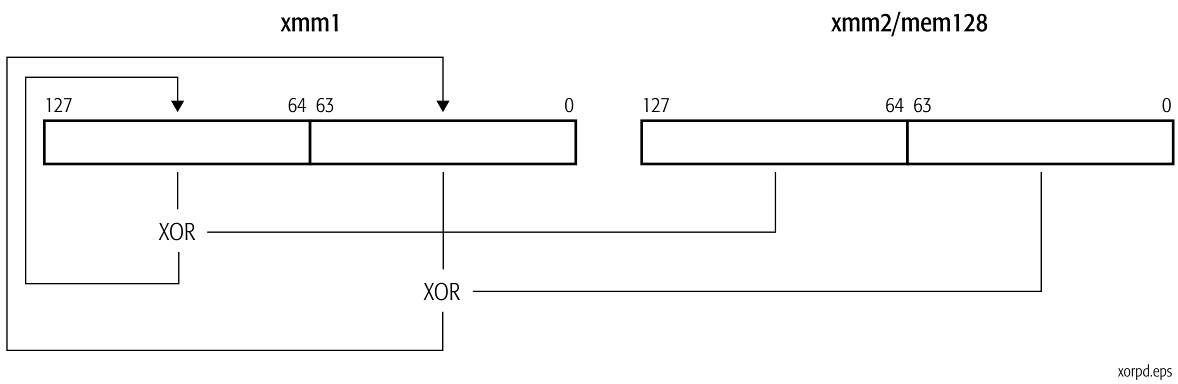
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

XORPD — Logical Bitwise Exclusive OR Packed Double-Precision Floating-Point

Performs a bitwise logical Exclusive OR of the two packed double-precision floating-point values in the first source operand and the corresponding two packed double-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The XORPD instruction is an SSE2 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
XORPD *xmm1*, *xmm2/mem128*	66 0F 57 /*r*	Performs bitwise logical XOR of two packed double-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPD, ORPS, XORPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

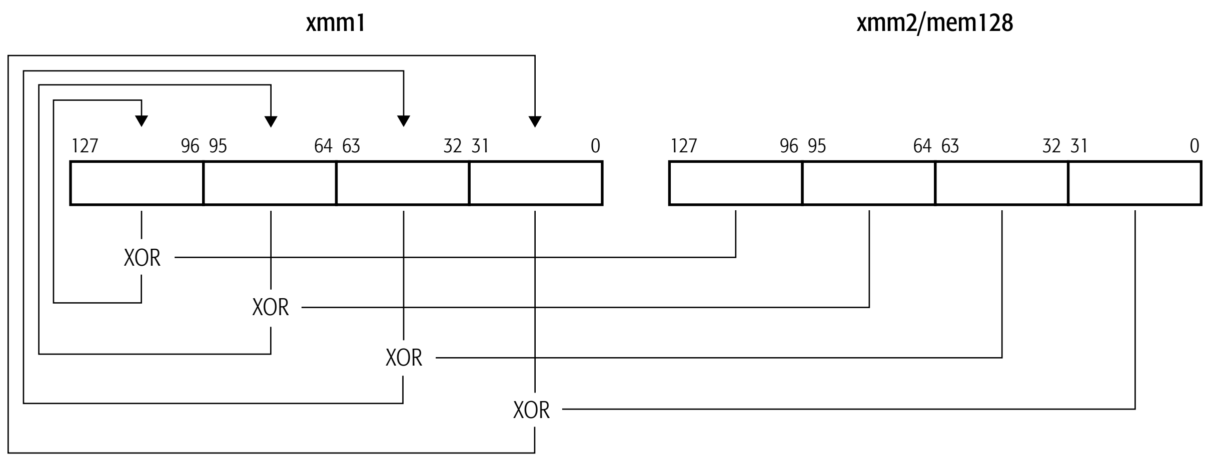
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE2 instructions are not supported, as indicated by EDX bit 26 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 is cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

XORPS Logical Bitwise Exclusive OR Packed Single-Precision Floating-Point

Performs a bitwise Exclusive OR of the four packed single-precision floating-point values in the first source operand and the corresponding four packed single-precision floating-point values in the second source operand and writes the result in the destination (first source). The first source/destination operand is an XMM register. The second source operand is another XMM register or 128-bit memory location.

The XORPS instruction is an SSE instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See “CPUID” in Volume 3.)

Mnemonic	Opcode	Description
XORPS *xmm1*, *xmm2/mem128*	0F 57 */r*	Performs bitwise logical XOR of four packed single-precision floating-point values in an XMM register and in another XMM register or 128-bit memory location and writes the result in the destination XMM register.



Related Instructions

ANDNPD, ANDNPS, ANDPD, ANDPS, ORPD, ORPS, XORPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE instructions are not supported, as indicated by EDX bit 25 of CPUID standard function 1.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 was cleared to 0.
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded the data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary.
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.

Index