AMD64 Technology

128-Bit SSE5 Instruction Set

Publication No.	Revision	Date
43479	**3.01**	**August 2007**

Advanced Micro Devices

Contents

Figures

Tables

Revision History

Date	Revision	Description
August 2007	3.01	Corrected the functional diagram for the CVTPS2PH instruction. Corrected typo on PTEST page.
August 2007	3.00	Initial public release.

Preface

About This Book

This book consists of documentation changes and additions to the multivolume *AMD64 Architecture Programmer's Manual*. The following 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 document is intended for all programmers writing application or system software for a processor that implements the AMD64 architecture.

1 New 128-Bit Instructions

This release of the AMD64 architecture introduces many new 128-bit instructions. The AMD64 128-bit media instructions are discussed in detail in the *AMD64 Architecture Programmer's Manual Volume 4: 128-Bit Media Instructions*, order# 26568. This document describes new instructions, including new three-operand instructions. Included are 23 base instructions, expanding to more than 100 total instructions, are designed to:

- Improve performance by increasing the work per instruction and
- Remove loads by reducing saving or reloading of register operands

New instructions include:

- Fused multiply accumulate (FMAC*xx)* instructions
- Integer multiply accumulate (IMAC, IMADC) instructions
- Permutation and conditional move instructions
- Vector compare and test instructions
- Precision control, rounding, and conversion instructions

Support for these instructions is provided by a new instruction encoding, which adds a third opcode byte (Opcode3). For the three- and four-operand instructions, a new *DREX* byte defines the destination register and provides the register extension information normally contained in a REX prefix. The REX prefix is not allowed with those instructions.

Support for the new instructions is indicated by ECX bit 11 (SSE5) as returned by CPUID function 8000_0001h. Attempting to execute these instructions causes a #UD exception if they are not present in the hardware.

1.1 New 128-Bit Media Instruction Format

This release introduces a new 128-bit media instruction format, which adds a third opcode byte, *Opcode3*. These instructions use opcodes 0F 24 00–FFh and 0F 25 00–FFh. Another new byte, the *DREX* byte, specifies the destination register and the REX extensions on the source operands. Instruction group 0F 25h is assigned to instructions that require a one-byte immediate operand, 0F 24h is assigned to instructions that do not. Prefixes 66h, F2h, and F3h can be used with opcode groups 0F 24h and 0F 25h to create new instruction maps. An invalid opcode exception results if a REX prefix is used with these opcodes.

Figure 1-1 on page 2 shows the byte order of the instruction format. The Opcode3 byte appears immediately after the two-byte Opcode, and the DREX byte appears immediately after the SIB byte (or ModRM byte, if there is no SIB byte).

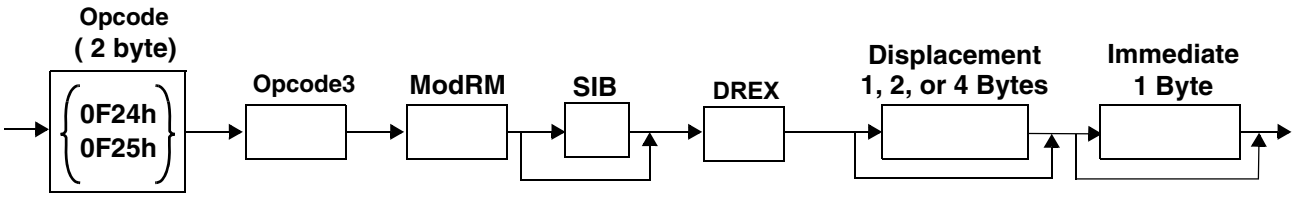


Figure 1-1. Instruction Byte-Order

1.1.1 Opcode3 Byte

The format of the Opcode3 byte is shown in Figure 1-2. A description of the fields is provided in Table 1-1.

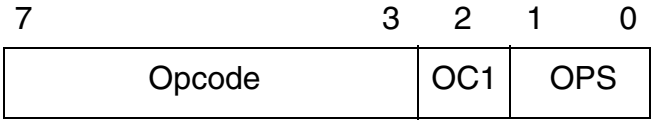


Figure 1-2. Opcode3 Byte Format

Table 1-1. Opcode3 Byte Fields

Field	Bit Position	Definition
Opcode	7-3	Provides additional opcode bits for the instructions
OC1	2	Operand Configuration Bit 1—Together with OC0, defines the order of the operands in the three and four operand instruction formats. For the four operand instruction format, see Table 1-4, “Operand Configurations for Four-Operand Instructions”, on page 4 for details. For the three operand instruction format, see Table 1-7, “Operand Configurations for Three Operand Instructions”, on page 6.
OPS	1-0	Operation Size - provides the size of the operation for both integer and floating-point. See Table 1-2 for details.

Table 1-2. Operation Size – OPS

Opcode3.OPS	Integer Operation	Floating-Point Operation
00	Byte	PS
01	Word	PD
10	Doubleword	SS
11	Quadword	SD

1.1.2 DREX Byte

The format of the DREX byte is shown in Figure 1-3. A description of the fields is provided in Table 1-3 below.

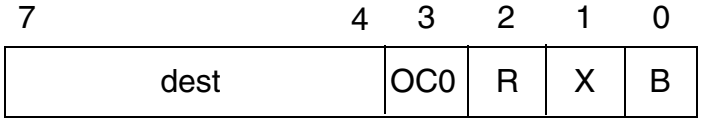


Figure 1-3. DREX Byte Format

Table 1-3. DREX Byte Fields

Mnemonic	Bit Position	Definition
DREX.dest	7-4	XMM destination register
DREX.OC0	3	Operand Configuration Bit 0 - Together with OC1, defines the order of the operands in the four operand instruction format. See Table 1-4, “Operand Configurations for Four-Operand Instructions”, on page 4 for details.
DREX.R	2	1-bit (high) extension of the ModRM *reg* field, thus permitting access to 16 XMM registers.
DREX.X	1	1-bit (high) extension of the SIB *index* field, thus permitting access to 16 registers.
DREX.B	0	1-bit (high) extension of the ModRM *r/m* field, SIB *base* field, or opcode *reg* field, thus permitting access to 16 registers.

Bits 7 and 2:0 are ignored in modes other than 64-bit.

1.2 Four-Operand 128-Bit Media Instructions

Some 128-bit media instructions have been derived from four-operand operations that require three input operands and one destination register. This is accomplished by mapping one of the three source operands to the destination operand by means of the DREX.dest field.

FMADDPS is an example of a four operand instruction:

FMADDPS *dest, src1, src2, src3; dest = src1 * src2 + src3*

The first operand is the destination operand and is an XMM register addressed by the 4-bit DREX.dest field. The second, third and fourth operands are source operands. One source operand is an XMM register addressed by the ModRM.reg field, another source operand is an XMM register or a memory operand addressed by the ModRM.r/m field, and another source operand is the same register as the destination register.

The OC1 and OC0 bits combine to determine which source operand is specified by which operand field in the opcode, as shown in Table 1-4 on page 4.

Instructions beginning with opcode bytes 0F 24h or 0F 25h take a DREX byte and do not use a REX prefix. The DREX.B, DREX.R and DREX.X bits are used to allow access to the REX registers.

Table 1-4. Operand Configurations for Four-Operand Instructions

OC[1:0]	dest	src1	src2	src3
00b	DREX.dest	DREX.dest	modrm.reg	modrm.r/m
01b	DREX.dest	DREX.dest	modrm.r/m	modrm.reg
10b	DREX.dest	modrm.reg	modrm.r/m	DREX.dest
11b	DREX.dest	modrm.r/m	modrm.reg	DREX.dest

The four operand instructions have opcodes in the 0F 24h and 0F 25h opcode pages. See Table 1-5.

Table 1-5. Four Operand Instruction Opcode Map

Operation	Opcode	Opcode3 [7:3]	Opcode3 [2] OC1	Opcode3 [1:0] OPS	DREX [3] OC0
COM[a]	0F 25 2C-2Fh	00101b	1b	OPS	0b
FMADD[a]	0F 24 00-07h	00000b	OC1	OPS	OC0
FMSUB[a]	0F 24 08-0Fh	00001b	OC1	OPS	OC0
FNMADD[a]	0F 24 10-17h	00010b	OC1	OPS	OC0
FNMSUB[a]	0F 24 18-1Fh	00011b	OC1	OPS	OC0
PCOM[a]	0F 25 4C-4Fh	01001b	1b	OPS	0b
PCOMU[a]	0F 25 6C-6Fh	01101b	1b	OPS	0b
PERMPS	0F 24 20,24h	00100b	OC1	00b	OC0
PERMPD	0F 24 21,25h	00100b	OC1	01b	OC0
PCMOV	0F 24 22,26h	00100b	OC1	10b	OC0
PPERM	0F 24 23,27h	00100b	OC1	11b	OC0
PMACSSWW	0F 24 85h	10000b	1b	01b	0b
PMACSWW	0F 24 95h	10010b	1b	01b	0b
PMACSSWD	0F 24 86h	10000b	1b	10b	0b
PMACSWD	0F 24 96h	10010b	1b	10b	0b
PMACSSDD	0F 24 8Eh	10001b	1b	10b	0b
PMACSDD	0F 24 9Eh	10011b	1b	10b	0b
PMACSSDQL	0F 24 87h	10000b	1b	11b	0b
PMACSDQL	0F 24 97h	10010b	1b	11b	0b
PMACSSDQH	0F 24 8Fh	10001b	1b	11b	0b
PMACSDQH	0F 24 9Fh	10011b	1b	11b	0b

Table 1-5. Four Operand Instruction Opcode Map (continued)

Operation	Opcode	Opcode3 [7:3]	Opcode3 [2] OC1	Opcode3 [1:0] OPS	DREX [3] OC0
PMADCSSWD	0F 24 A6h	10100b	1b	10b	0b
PMADCSWD	0F 24 B6h	10110b	1b	10b	0b

a. Indicates four instruction variants (_PS, _PD, _SS and _SD) specified by the OPS field.

1.2.1 NaN Results on FMAC Instructions

When a three source operand floating-point operation such as FMADDPS produces a QNaN result, its value is determined by the rules in Table 1-6.

Table 1-6. NaN Results for SRC1 * SRC2 + SRC3

SRC1	SRC2	SRC3	Result
SNaN1	any	any	QNaN1, IE
QNaN1	SNaN2	any	QNaN1, IE
QNaN1	any	SNaN3	QNaN1, IE
!NaN	SNaN2	any	QNaN2, IE
!NaN	QNaN2	SNaN3	QNaN2, IE
!NaN	!NaN	SNaN3	QNaN3, IE
QNaN1	!SNaN2	!SNaN3	QNaN1
!NaN	QNaN2	!SNaN3	QNaN2
!NaN	!NaN	QNaN3	QNaN3
zero	infinity	!NaN	QNaN(indefinite), IE
infinity	zero	!NaN	QNaN(indefinite), IE
product=+infinity[a]		-infinity	QNaN(indefinite), IE
product=-infinity[a]		+infinity	QNaN(indefinite), IE

a. The +infinity or -infinity product requires one source operand to be infinity and the other source operand to be a valid non-zero value.

QNaN—quiet NaN
SNaN—signaling NaN
!SNaN—a number that does not represent a signaling NaN.
!NaN—either normal, denormal (including zero) or infinity
IE—Invalid-operation exception

1.3 Three-Operand 128-Bit Media Instructions

Some instructions have two source operands and a destination operand.

PROTB is an example of a three operand instruction:

PROTB dest, src, count dest = src <</>> count

The first operand is the destination operand, and is an XMM register addressed by the 4-bit DREX.dest field. The second and third operands are source operands. One source operand is an XMM register addressed by the ModRM.reg field, the other source operand is an XMM register or memory operand addressed by the ModRM.r/m field.

In the three-operand format the OC1 bit is used as an extension to the opcode. The OC0 bit determines which source operand is specified by which operand field, as shown in Table 1-7.

The instructions with a DREX byte do not use the REX prefix. The DREX.R, DREX.B and DREX.X bits are used to allow access to the REX registers.

Table 1-7. Operand Configurations for Three Operand Instructions

OC0	dest	src	count
0b	drex.dest	modrm.reg	modrm.r/m
1b	drex.dest	modrm.r/m	modrm.reg

The three operand instructions have opcodes in the 0F 24h page. See Table 1-8.

Table 1-8. Three Operand Instruction Opcode Map

Operation	Opcode	Opcode3[7:3]	Opcode3[2] OC1	Opcode3[1:0] OPS	DREX[3] OC0
PROT[a]	0F 24 40-43h	01000b	0b	OPS	OC0
PSHL[a]	0F 24 44-47h	01000b	1b	OPS	OC0
PSHA[a]	0F 24 48-4Bh	01001b	0b	OPS	OC0

a. Indicates four instruction variants (_B, _W, _D and _Q) specified by the OPS field.

Note that there is only one operand configuration for the COM, PCOM and PCOMU instructions. The OC0 bit is zero.

1.4 Other 128-Bit Media Instructions

Other instructions use the normal two byte operand assignment. The first instruction operand (xmm1) is the destination, addressed by the ModRM.reg field. The second operand (xmm2/mem128) is either an XMM register or memory operand, as determined by the ModRM and SIB.

CVTPH2PS is an example of a two operand instruction.

CVTPH2PS *xmm1*, *xmm2*/*mem64*

The new instructions with one or two operands are assigned to two-byte opcodes 0F 3Ah (ROUND), 0F 7Ah, 0F 7Bh (PROT*x*) and 0F 38h (PTEST). See Table 1-9 on page 7.

Table 1-9. One/Two Operand Instruction Opcode Map

Operation	Opcode	Opcode3[7:3]	Opcode3[2] OC1	Opcode3[1:0] OPS
FRCZ[b]	0F 7A 10-13h	00010b	0b	OPS
CVTPH2PS	0F 7A 30h	00110b	0b	00b
CVTPS2PH	0F 7A 31h	00110b	0b	01b
PHADDBW	0F 7A 41h	01000b	0b	01b
PHADDBD	0F 7A 42h	01000b	0b	10b
PHADDBQ	0F 7A 43h	01000b	0b	11b
PHADDWD	0F 7A 46h	01000b	1b	10b
PHADDWQ	0F 7A 47h	01000b	1b	11b
PHADDDQ	0F 7A 4Bh	01001b	0b	11b
PHADDUBW	0F 7A 51h	01010b	0b	01b
PHADDUBD	0F 7A 52h	01010b	0b	10b
PHADDUBQ	0F 7A 53h	01010b	0b	11b
PHADDUWD	0F 7A 56h	01010b	1b	10b
PHADDUWQ	0F 7A 57h	01010b	1b	11b
PHADDUDQ	0F 7A 5Bh	01011b	0b	11b
PHSUBBW	0F 7A 61h	01100b	0b	01b
PHSUBWD	0F 7A 62h	01100b	0b	10b
PHSUBDQ	0F 7A 63h	01100b	0b	11b
PROT[a]	0F 7B 40-43h	01000b	0b	OPS
PTEST	66 0F 38 17	00010b	1b	11b
ROUND[b]	66 0F 3A 08-0B	00001b	0b	OPS

a. Indicates four instruction variants (_B, _W, _D and _Q) specified by the OPS field.
b. Indicates four instruction variants (_PS, _PD, _SS and _SD) specified by the OPS field.

1.5 16-Bit Floating-Point Data Type

SSE5 introduces a new 16-bit floating-point data type and two instructions (CVTPS2PH and CVTPH2PS) to convert 16-bit floating-point values to and from single-precision format.

The 16-bit floating-point data type, shown in Figure 1-4 on page 8, includes a 1-bit sign, a 5-bit exponent with a bias of 15 and a 10-bit significand. The integer bit is implied, making a total of 11 bits in the significand. The value of the integer bit can be inferred from the number encoding. Table 1-10 on page 8 shows the floating-point encodings of supported numbers and non-numbers.

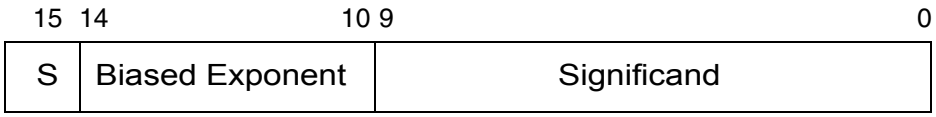


Figure 1-4. 16-Bit Floating-Point Data Type

Table 1-10. Supported 16-Bit Floating-Point Encodings

Sign	Bias Exponent	Significand[a]	Classification	
0	1 1111	1.01 1111 1111 to 1.00 0000 0001	Positive Non-Number	SNaN
0	1 1111	1.11 1111 1111 to 1.10 0000 0001		QNaN
0	1 1111	1.00 0000 0000	Positive Floating-Point Numbers	Positive Infinity
0	1 1110 to 0 0001	1.11 1111 1111 to 1.00 0000 0000		Positive Normal
0	0 0000	0.11 1111 1111 to 0.00 0000 0001		Positive Denormal
0	0 0000	0.00 0000 0000		Positive Zero
1	0 0000	0.00 0000 0000	Positive Floating-Point Numbers	Negative Zero
1	0 0000	0.00 0000 0001 to 0.11 1111 1111		Negative Denormal
1	0 0001 to 1 1110	1.00 0000 0000 to 1.11 1111 1111		Negative Normal
1	1 1111	1.00 0000 0000		Negative Infinity
1	1 1111	1.00 0000 0001 to 1.01 1111 1111	Negative Non-Number	SNaN
1	1 1111	1.10 0000 0001 to 1.11 1111 1111		QNaN

a. The “1.” and “0.” prefixes represent the implicit integer bit.

1.6 Floating Point Multiply and Add/Subtract

The combined operation of the floating-point (negative) multiplication and addition/subtraction operations is shown in Figure 1-5. The negative multiply instructions apply the negation to the results of the multiplication before applying the addition or subtraction operation.

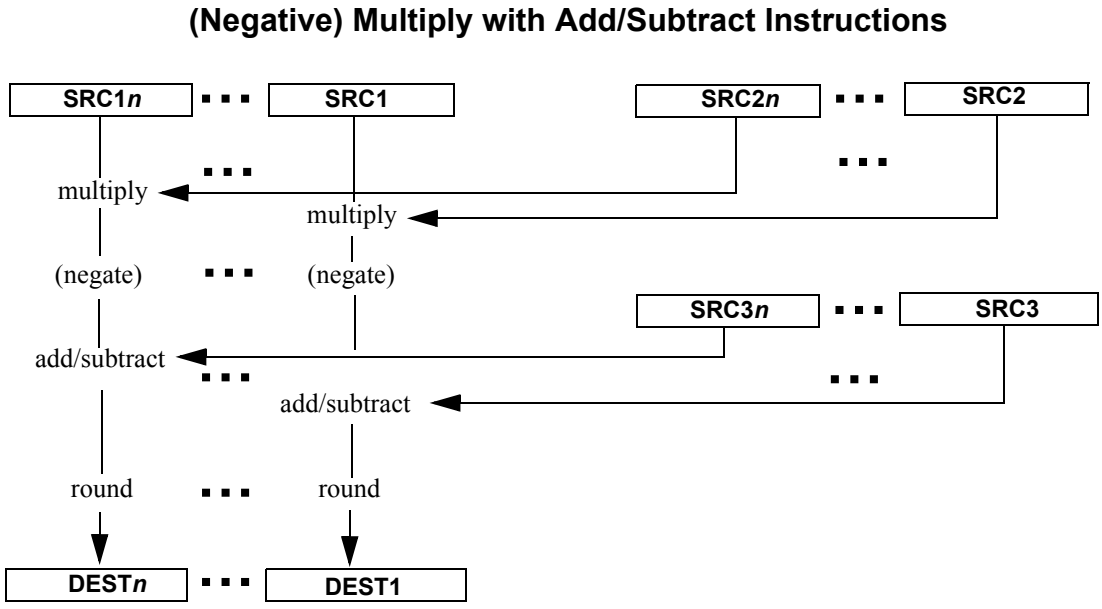


Figure 1-5. Operation of Multiplication with Addition/Subtraction Instructions

The SSE5 instructions set includes the following combined multiply with add/subtract instructions. Note that scalar instructions only operate on the lowest element of the specified size in the source and destination registers; the contents of the upper elements of the source and destination registers are unaffected by the operation.

- FMADDPS—Multiply and Add Packed Single-Precision Floating Point
- FMADDPD—Multiply and Add Packed Double-Precision Floating Point
- FMADDSS—Multiply and Add Scalar Single-Precision Floating Point
- FMADDSD—Multiply and Accumulate Scalar Double-Precision Floating Point
- FMSUBPS—Multiply and Subtract Packed Single-Precision Floating-Point
- FMSUBPD—Multiply and Subtract Packed Double-Precision Floating-Point
- FMSUBSS—Multiply and Subtract Scalar Single-Precision Floating-Point
- FMSUBSD—Multiply and Subtract Scalar Double-Precision Floating-Point
- FNMADDPS—Negative Multiply and Add Packed Single-Precision Floating-Point
- FNMADDPD—Negative Multiply and Add Packed Double-Precision Floating-Point

- FNMADDSS—Negative Multiply and Add Scalar Single-Precision Floating-Point
- FNMADDSD—Negative Multiply and Add Scalar Double-Precision Floating-Point
- FNMSUBPS—Negative Multiply and Subtract Packed Single-Precision Floating-Point
- FNMSUBPD—Negative Multiply and Subtract Packed Double-Precision Floating-Point
- FNMSUBSS—Negative Multiply and Subtract Scalar Single-Precision Floating-Point
- FNMSUBSD—Negative Multiply and Subtract Scalar Double-Precision Floating-Point

1.7 Integer Multiply (Add) and Accumulate Instructions

The multiply and accumulate and multiply, add and accumulate instructions operate on and produce packed signed integer values. These instructions allow the accumulation of results from (possibly) many iterations of similar operations without a separate intermediate addition operation to update the accumulator register. The accumulator is both a source (*src3*) and a destination register (*dest*)—it is an XMM register addressed by the DREX.dest field.

1.7.1 Saturation

Some instructions limit the result of an operation to the maximum or minimum value representable by the data type of the destination—an operation known as *saturation*. Many of the integer multiply and accumulate instructions saturate the cumulative results of the multiplication and addition (accumulation) operations before writing the final results to the destination (accumulator) register.

Note, however, that not all multiply and accumulate instructions saturate results. (For further discussion of saturation, see the *AMD64 Architecture Programmer's Manual Volume 1: Application Programming*, order# 24592.)

1.7.2 Multiply and Accumulate Instructions

The operation of a typical SSE5 integer multiply and accumulate instruction is shown in Figure 1-6 on page 11.

The multiply and accumulate instructions operate on and produce packed signed integer values. These instructions first multiply the value in the first source operand by the corresponding value in the second source operand. Each signed integer product is then added to the corresponding value in the third source operand, which is the accumulator and is identical to the destination operand. The results may or may not be saturated prior to being written to the destination register, depending on the instruction.

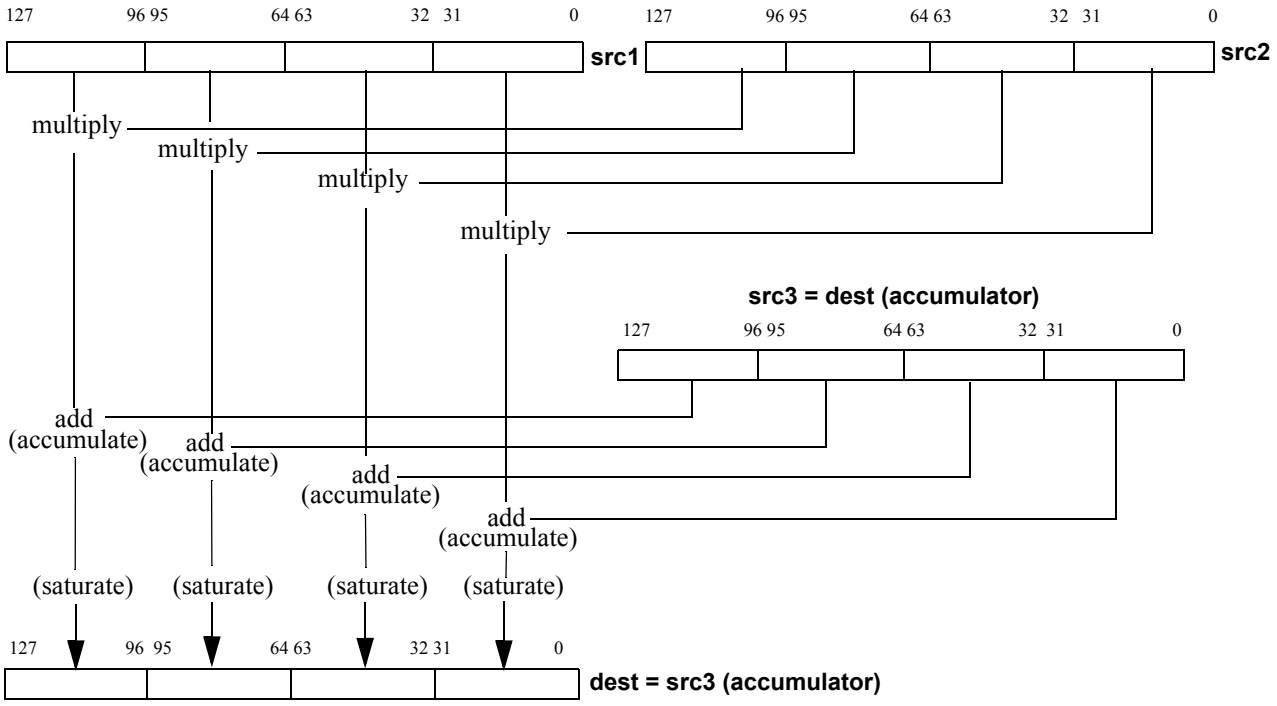


Figure 1-6. Operation of Multiply and Accumulate Instructions

The SSE5 instruction set provides the following integer multiply and accumulate instructions.

- PMACSSWW—Packed Multiply Accumulate Signed Word to Signed Word with Saturation
- PMACSWW—Packed Multiply Accumulate Signed Word to Signed Word
- PMACSSWD—Packed Multiply Accumulate Signed Word to Signed Doubleword with Saturation
- PMACSWD—Packed Multiply Accumulate Signed Word to Signed Doubleword
- PMACSSDD—Packed Multiply Accumulate Signed Doubleword to Signed Doubleword with Saturation
- PMACSDD—Packed Multiply Accumulate Signed Doubleword to Signed Doubleword
- PMACSSDQL—Packed Multiply Accumulate Signed Low Doubleword to Signed Quadword with Saturation
- PMACSSDQH—Packed Multiply Accumulate Signed High Doubleword to Signed Quadword with Saturation
- PMACSDQL—Packed Multiply Accumulate Signed Low Doubleword to Signed Quadword
- PMACSDQH—Packed Multiply Accumulate Signed High Doubleword to Signed Quadword

1.7.3 SSE5 Integer Multiply, Add and Accumulate Instructions

The operation of the multiply, add and accumulate instructions is illustrated in Figure 1-7.

The multiply, add and accumulate instructions first multiply each packed signed integer value in the first source operand by the corresponding packed signed integer value in the second source operand. The odd and even adjacent resulting products are then added. Each resulting sum is then added to the corresponding packed signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register addressed by the DREX.dest field.

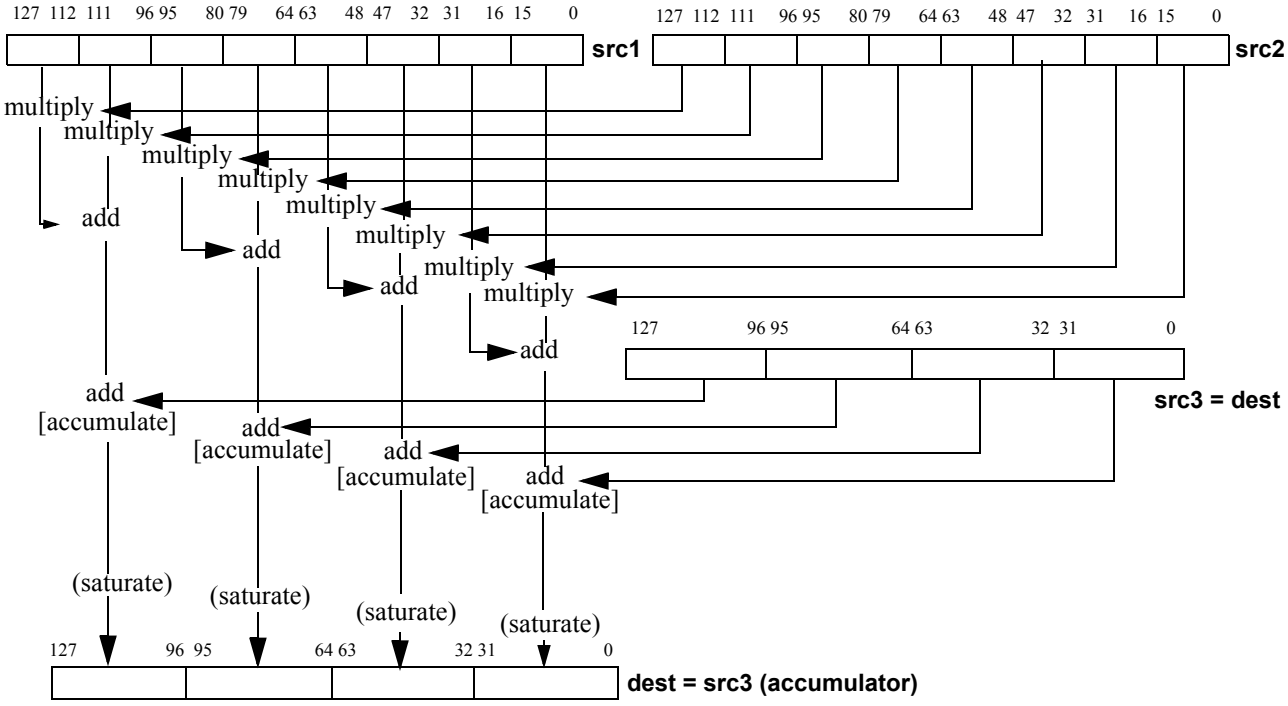


Figure 1-7. Operation of Multiply, Add and Accumulate Instructions

The SSE5 instruction set provides the following integer multiply, add and accumulate instructions.

- PMADCSSWD—Packed Multiply Add and Accumulate Signed Word to Signed Doubleword with Saturation
- PMADCSWD—Packed Multiply Add and Accumulate Signed Word to Signed Doubleword

1.8 Packed Integer Horizontal Add and Subtract

The packed horizontal add and subtract signed byte instructions successively add adjacent pairs of signed integer values from the second source XMM register or 128-bit memory operand and pack the (sign-extended) integer result of each addition in the destination (first source).

- PHADDBW—Packed Horizontal Add Signed Byte to Signed Word
- PHADDBD—Packed Horizontal Add Signed Byte to Signed Doubleword
- PHADDBQ—Packed Horizontal Add Signed Byte to Signed Quadword
- PHADDDQ—Packed Horizontal Add Signed Doubleword to Signed Quadword
- PHADDUBW—Packed Horizontal Add Unsigned Byte to Word
- PHADDUBD—Packed Horizontal Add Unsigned Byte to Doubleword
- PHADDUBQ—Packed Horizontal Add Unsigned Byte to Quadword
- PHADDUWD—Packed Horizontal Add Unsigned Word to Doubleword
- PHADDUWQ—Packed Horizontal Add Unsigned Word to Quadword
- PHADDUDQ—Packed Horizontal Add Unsigned Doubleword to Quadword
- PHADDWD—Packed Horizontal Add Signed Word to Signed Doubleword
- PHADDWQ—Packed Horizontal Add Signed Word to Signed Quadword
- PHSUBBW—Packed Horizontal Subtract Signed Byte to Signed Word
- PHSUBWD—Packed Horizontal Subtract Signed Word to Signed Doubleword
- PHSUBDQ—Packed Horizontal Subtract Signed Doubleword to Signed Quadword

1.9 Vector Conditional Moves

SSE5 instructions include four vector conditional moves instructions:

- PCMOV—Vector Conditional Moves
- PPERM—Packed Permute Bytes
- PERMPS—Permute and Modify Single-Precision Floating Point
- PERMPD—Permute Double-Precision Floating Point

The PCMOV instruction implements the C/C++ language ternary ‘?’ operator. This instruction operates on individual bits and requires a bitwise predicate in one XMM register and the two source operands in two more XMM registers.

The PPERM instruction performs vector permutation on a packed array of 32 bytes. The PPERM instruction replaces some or all of its destination bytes with 0x00, 0xFF, or one of the 32 bytes of the packed array. A byte selected from the array may have an additional operation such as NOT or bit reversal applied to it, before it is written to the destination. The action for each destination byte is determined by a corresponding control byte.

PERMP*x* instructions provides a superset of the SHUFPS instruction. This instruction performs a permutation operation on an array of eight single-precision or four double-precision floating-point values, optionally followed by an additional operation (ABS, NEG, NEGABS, set-to-constant (0.0, -1.0, 1.0, PI)).

1.10 Packed Integer Rotates and Shifts

These instructions rotate/shift the elements of the vector in the first source XMM or 128-bit memory operand by the amount specified by a control byte. The rotates and shifts differ in the way they handle the control byte.

1.10.1 Packed Integer Shifts

The packed integer shift instructions shift each element of the vector in the first source XMM or 128-bit memory operand by the amount specified by a control byte contained in the least significant byte of the corresponding element of the second source operand. The result of each shift operation is returned in the destination XMM register. This allows load-and-rotate from memory operations. The SSE5 instruction set provides the following packed integer shift instructions:

- PSHLB—Packed Shift Logical Bytes
- PSHLW—Packed Shift Logical Words
- PSHLD—Packed Shift Logical Doublewords
- PSHLQ—Packed Shift Logical Quadwords
- PSHAB—Packed Shift Arithmetic Bytes
- PSHAW—Packed Shift Arithmetic Words
- PSHAD—Packed Shift Arithmetic Doublewords
- PSHAQ—Packed Shift Arithmetic Quadwords

1.10.2 Packed Integer Rotate

There are two variants of the packed integer rotate instructions. The first is identical to that described above (see “Packed Integer Shifts”). In the second variant, the control byte is supplied by an immediate operand that determines the identical amount to rotate for every element in the first source operand. The SSE5 instruction set provides the following packed integer rotate instructions:

- PROTB—Packed Rotate Bytes
- PROTW—Packed Rotate Words
- PROTD—Packed Rotate Doublewords
- PROTQ—Packed Rotate Quadwords

1.11 Floating Point Comparison and Predicate Generation

The SSE5 comparison instructions compare floating-point or integer values in the first source XMM register with corresponding floating point or integer values in the second source XMM register or 128-bit memory. The type of comparison is specified by the immediate-byte operand. The resulting predicate is placed in the destination XMM register. If the condition is true, all bits in the corresponding field in the destination register are set to 1s; otherwise all bits in the field are set to 0s.

1.11.1 Floating-Point Comparison Operations

The type of comparison of the floating-point comparison operation is specified by the *four* low-order bits of the immediate-byte operand. If the condition is true, all corresponding field in the destination will be set to all 1s; otherwise it will be set to all 0s.

Comparisons can be ordered or unordered. Ordered comparisons return TRUE only if both operands are valid numbers and the numbers have the relation specified by the type of comparison; they are FALSE otherwise.

Unordered comparisons return TRUE if one of the operands is a NaN or the numbers have the relation specified by the type of comparison; otherwise, they are FALSE.

- COMPS—Compare Vector Single-Precision Floating Point
- COMPD—Compare Vector Double-Precision Floating Point
- COMSS—Compare Scalar Single-Precision Floating Point
- COMSD—Compare Scalar Double-Precision Floating Point

1.11.2 Integer Comparison and Predicate Generation

The integer comparison and predicate generation instructions compare corresponding packed unsigned bytes in the first and second source operands and write the result of each comparison in the corresponding byte of the destination. The result of each comparison is a value of all 1s (TRUE) or all 0s (FALSE). The type of comparison is specified by the three low-order bits of the immediate-byte operand. The SSE5 instruction set provides the following integer comparison instructions.

- PCOMUB—Compare Vector Unsigned Bytes
- PCOMUW—Compare Vector Unsigned Words
- PCOMUD—Compare Vector Unsigned Doublewords
- PCOMUQ—Compare Vector Unsigned Quadwords
- PCOMB—Compare Vector Signed Bytes
- PCOMW—Compare Vector Signed Words
- PCOMD—Compare Vector Signed Doublewords
- PCOMQ—Compare Vector Signed Quadwords

1.12 Test Instruction

The PTEST instruction performs a bitwise logical AND between the source XMM register or 128-bit memory location and destination XMM register. The ZF flag is set to 1 if all bit positions that are set to 1 in the mask operand are set to 0 in the source operand; otherwise, ZF is cleared. The CF flag is set to 1 if all bit positions specified in the mask operand are set to 1 in the source operand; otherwise, CF is cleared.

- PTEST—Predicate Test Register

1.13 Precision Control and Rounding

The precision control and rounding instructions can move (from memory) and round data with a single instruction. The result of _PD and _PS instructions is a vector of floating-point numbers. The result of _SD and _SS instructions is always a scalar floating-point number. SSE5 provides the following precision control and rounding instructions:

- FRCZPD—Extract Fraction Packed Double-Precision Floating-Point
- FRCZPS—Extract Fraction Packed Single-Precision Floating-Point
- FRCZSD— Extract Fraction Scalar Double-Precision Floating-Point
- FRCZSS— Extract Fraction Scalar Single-Precision Floating Point
- ROUNDPD—Round Packed Double-Precision Floating-Point
- ROUNDPS—Round Packed Single-Precision Floating-Point
- ROUNDSD— Round Scalar Double-Precision Floating-Point
- ROUNDSS—Round Scalar Single-Precision Floating-Point Convert

The FRCZPD and FRCZPS instructions extract the fractional portions of a vector of double-/single-precision floating-point values in an XMM register or a 128-bit memory location and write the results in the corresponding field in the destination register.

The FRCZSS and FRCZSD instructions extract the fractional portion of the single-/double-precision scalar floating-point value in and XMM register or 128-bit memory location and writes the results in the corresponding field in the destination register. The upper fields of the destination register are unaffected by the operation.

The ROUNDPD and ROUNDPD instructions round the double-/single-precision floating-point values in an XMM register or a 128-bit memory location to the nearest integer, as determined by the rounding mode specified by the 8-bit immediate control byte and write the floating-point results in the corresponding fields in a destination XMM register.

The ROUNDSD and ROUNDSS instructions round the double-/single-precision scalar floating-point value in the low position of an XMM register or a 64-bit memory location to the nearest integer, as determined by the rounding mode specified by the 8-bit immediate control byte and writes the results as a double-precision floating-point value in the low 64 bits of the destination XMM register. The upper fields of the destination register are unaffected by the operation.

1.14 Convert

Two SSE5 instructions are provided to move data from/to memory and convert a single-precision floating point to 16-Bit floating-point or vice versa in one instruction. (See Section 1.5, “16-Bit Floating-Point Data Type,” on page 7.)

- CVTPH2PS—Convert 16-Bit Floating-Point to Single-Precision Floating Point
- CVTPS2PH—Convert Single-Precision Floating-Point to 16-Bit Floating Point

2 SSE5 128-Bit Media Instructions

The following section describes the complete set of SSE5 128-media instructions. Instructions are listed alphabetically by mnemonic.

2.1 Notation

The notation used to denote the size and type of source and destination operands in both mnemonics and opcodes is discussed in detail in Section 2.5, “Notation,” on page 37 in the *AMD64 Architecture Programmer’s Manual Volume 3: General Purpose and System Instructions*. Mnemonic conventions that are idiosyncratic to the SSE5 instruction set have been included in *Chapter 1, “New 128-Bit Instructions”*, in this document.

2.1.1 Opcode Syntax

In addition to the opcode notational conventions specified in Section 2.5.2, “Opcode Syntax,” on page 39 in the *AMD64 Architecture Programmer’s Manual Volume 3: General Purpose and System Instructions*, the SSE5 instruction set requires the following notation to indicates the value of the DREX.OC0 bit:

/drex0—Indicates a DREX byte, with the OC0 bit cleared to zero.

/drex1—Indicates a DREX byte, with the OC0 bit set to one.

2.2 Instruction Reference

COMPD Compare Vector Double-Precision Floating-Point

Compares each of the two double-precision floating-point values in the first source operand with the corresponding two double-precision floating-point values in the second source operand and writes the result of each comparison in the corresponding 64 bits of the destination. The result of each comparison is a 64-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the four low-order bits of the immediate-byte operand, as shown in the following table.

COM Immediate Operand

Immediate Operand Byte				
Bits	Descriptions			
7:4	0000b			
3:0	cond – Defines the comparison operation performed on the selected operand.			
	cond	**Comparison Operation**	**Result if NaN Operand**	**QNaN Operand Causes Invalid Operation Exception**
	0000	Ordered and Equal	FALSE	No
	0001	Ordered and Less Than	FALSE	Yes
	0010	Ordered and Not Greater Than	FALSE	Yes
	0011	Unordered	TRUE	No
	0100	Unordered or Not Equal	TRUE	No
	0101	Unordered or Not Less Than	TRUE	Yes
	0110	Unordered or Greater Than	TRUE	Yes
	0111	Ordered	FALSE	No
	1000	Unordered or Equal	TRUE	No
	1001	Unordered or Less Than	TRUE	No
	1010	Unordered or Not Greater Than	TRUE	No
	1011	False	FALSE	No
	1100	Ordered and Not Equal	FALSE	No
	1101	Ordered and Not Less Than	FALSE	No
	1110	Ordered and Greater Than	FALSE	No
	1111	True	TRUE	No

There are two types of comparisons, ordered and unordered. Ordered comparison operations return TRUE only if both operands are valid numbers and the numbers have the relation specified by the type of comparison and FALSE otherwise. Unordered comparison operations return TRUE if one of the operands is a NaN, or the numbers have the relation specified by the type of comparison; and FALSE

otherwise. The “True” and “False” operations return all 1s and all 0s, respectively, regardless of whether any of the source operands is a NaN.

QNaN operands generate an Invalid Operation Exception only if the comparison type is “Less than (or Equal)” and “Greater than (or Equal)”. SNaN operands generate an Invalid Operation (IE) exception for all operations, including “True” and “ False”.

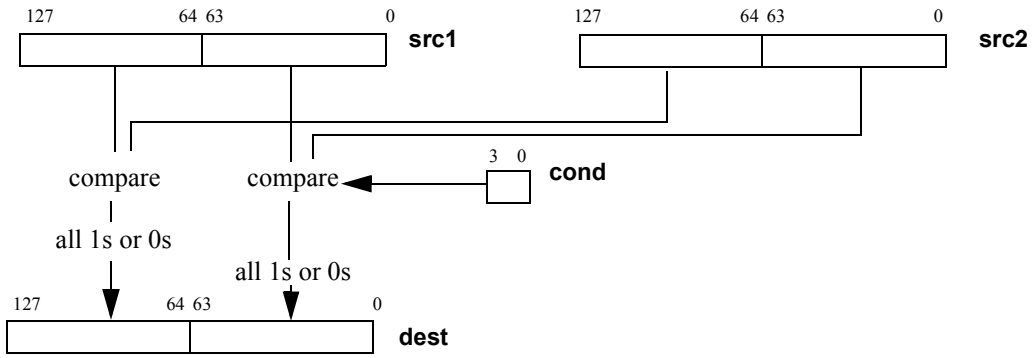
The COMPD instruction requires four operands:

COMPD dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The COMPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
COMPD *xmm1, xmm2, xmm3/mem128, imm8*	0F 25 2D /r /drex0 ib	Compares two packed double-precision floating-point values in XMM2 register by XMM3 register or 128-bit memory location and writes 64 bits of all 1s (TRUE) or all 0s (FALSE) in the destination (XMM1 register).



Related Instructions

COMPS, COMSS, COMSD, CMPPD, CMPPS, CMPSS, CMPSD, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
															M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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 on page 18).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

COMPS Compare Vector Single-Precision Floating-Point

Compares each of the four single-precision floating-point values in the first source operand with the corresponding four single-precision floating-point values in the second source operand and writes the result of each comparison in the corresponding 32 bits of the destination. The result of each comparison is a 32-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the four low-order bits of the immediate-byte operand, as shown in the following table.

COM Immediate Operand

Immediate Operand Byte				
Bits	Descriptions			
7:4	0000b			
3:0	cond – Defines the comparison operation performed on the selected operand.			
	cond	**Comparison Operation**	**Result if NaN Operand**	**QNaN Operand Causes Invalid Operation Exception**
	0000	Ordered and Equal	FALSE	No
	0001	Ordered and Less Than	FALSE	Yes
	0010	Ordered and Not Greater Than	FALSE	Yes
	0011	Unordered	TRUE	No
	0100	Unordered or Not Equal	TRUE	No
	0101	Unordered or Not Less Than	TRUE	Yes
	0110	Unordered or Greater Than	TRUE	Yes
	0111	Ordered	FALSE	No
	1000	Unordered or Equal	TRUE	No
	1001	Unordered or Less Than	TRUE	No
	1010	Unordered or Not Greater Than	TRUE	No
	1011	False	FALSE	No
	1100	Ordered and Not Equal	FALSE	No
	1101	Ordered and Not Less Than	FALSE	No
	1110	Ordered and Greater Than	FALSE	No
	1111	True	TRUE	No

Ordered comparison operations return TRUE only if both operands are valid numbers and the numbers have the relation specified by the type of comparison and FALSE otherwise. Unordered comparison operations return TRUE if one of the operands is a NaN, or the numbers have the relation specified by the type of comparison; and FALSE otherwise. The “True” and “False” operations return all 1s and all 0s, respectively, regardless of whether any of the source operands is a NaN.

QNaN operands generate an Invalid Operation Exception only if the comparison type is "(Not) Less than (or Equal)". SNaN operands generate an Invalid Operation (IE) exception for all operations, including "True" and " False".

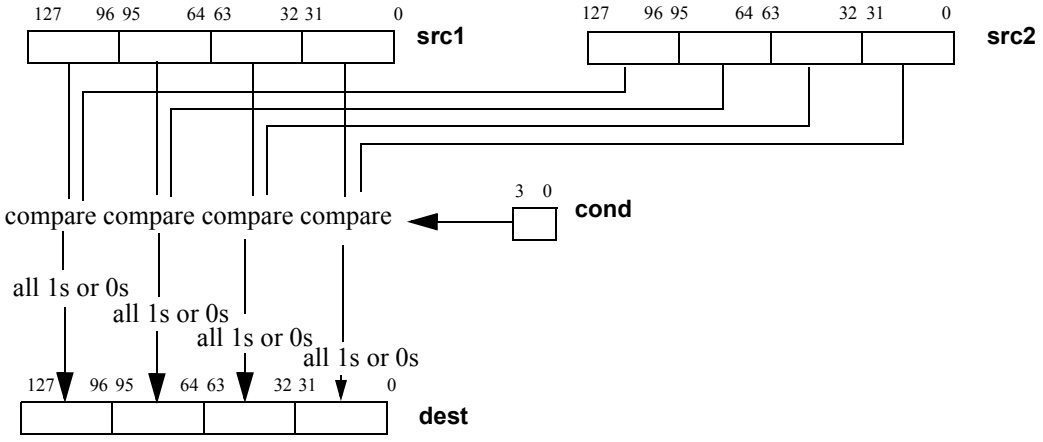
The COMPS instruction requires four operands:

COMPS dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The COMPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
COMPS *xmm1*, *xmm2, xmm3/mem128, imm8*	0F 25 2C /r /drex0 ib	Compares four packed single-precision floating-point values in XMM2 register by XMM3 register or 128-bit memory location and writes 32 bits of all 1s (TRUE) or all 0s (FALSE) in the destination (XMM1 register).



Related Instructions

COMPD, COMSS, COMSD, CMPPD, CMPPS, CMPSS, CMPSD, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
															M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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 on page 18).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

COMSD 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 of the comparison in the low-order 64 bits of the destination. The high-order quadword of the destination is cleared to 0s. The result of the comparison is a 64-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the four low-order bits of the immediate-byte operand, as shown in the following table.

COM Immediate Operand

Immediate Operand Byte				
Bits	Descriptions			
7:4	0000b			
3:0	cond – Defines the comparison operation performed on the selected operand.			
	cond	**Comparison Operation**	**Result if NaN Operand**	**QNaN Operand Causes Invalid Operation Exception**
	0000	Ordered and Equal	FALSE	No
	0001	Ordered and Less Than	FALSE	Yes
	0010	Ordered and Not Greater Than	FALSE	Yes
	0011	Unordered	TRUE	No
	0100	Unordered or Not Equal	TRUE	No
	0101	Unordered or Not Less Than	TRUE	Yes
	0110	Unordered or Greater Than	TRUE	Yes
	0111	Ordered	FALSE	No
	1000	Unordered or Equal	TRUE	No
	1001	Unordered or Less Than	TRUE	No
	1010	Unordered or Not Greater Than	TRUE	No
	1011	False	FALSE	No
	1100	Ordered and Not Equal	FALSE	No
	1101	Ordered and Not Less Than	FALSE	No
	1110	Ordered and Greater Than	FALSE	No
	1111	True	TRUE	No

There are two types of comparisons, ordered and unordered. Ordered comparison operations return TRUE only if both operands are valid numbers and the numbers have the relation specified by the type of comparison and FALSE otherwise. Unordered comparison operations return TRUE if one of the operands is a NaN, or the numbers have the relation specified by the type of comparison; and FALSE otherwise. The “True” and “False” operations return all 1s and all 0s, respectively, regardless of whether any of the source operands is a NaN.

QNaN operands generate an Invalid Operation Exception only if the comparison type is “Less than (or Equal)” and “Greater than (or Equal)”. SNaN operands generate an Invalid Operation (IE) exception for all operations, including “True” and “ False”.

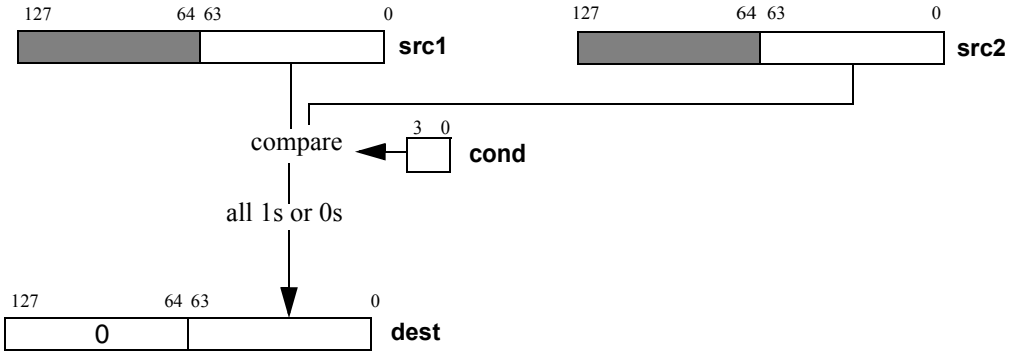
The COMSD instruction requires four operands:

COMSD dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The COMSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
COMSD *xmm1*, *xmm2, xmm3/mem64, imm8*	0F 25 2F /r /drex0 ib	Compares the low-order double-precision floating-point value in XMM2 register by the low-order double-precision floating-point value in XMM3 register or 64-bit memory location and writes 64 bits of all 1s (TRUE) or all 0s (FALSE) in the low-order quadword in the destination (XMM1 register).



Related Instructions

COMPS, COMPD, COMSS, CMPPD, CMPPS, CMPSS, CMPSD, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
															M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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	A source operand was a QNaN value and the comparison does not allow QNaN values (refer to Table on page 18).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

COMSS 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 of the comparison in the low-order 32 bits of the destination. The three high-order doublewords of the destination are cleared to 0s. The result of the comparison is a 32-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the four low-order bits of the immediate-byte operand, as shown in the following table.

COM Immediate Operand

Immediate Operand Byte				
Bits	Descriptions			
7:4	0000b			
3:0	cond – Defines the comparison operation performed on the selected operand.			
	cond	**Comparison Operation**	**Result if NaN Operand**	**QNaN Operand Causes Invalid Operation Exception**
	0000	Ordered and Equal	FALSE	No
	0001	Ordered and Less Than	FALSE	Yes
	0010	Ordered and Not Greater Than	FALSE	Yes
	0011	Unordered	TRUE	No
	0100	Unordered or Not Equal	TRUE	No
	0101	Unordered or Not Less Than	TRUE	Yes
	0110	Unordered or Greater Than	TRUE	Yes
	0111	Ordered	FALSE	No
	1000	Unordered or Equal	TRUE	No
	1001	Unordered or Less Than	TRUE	No
	1010	Unordered or Not Greater Than	TRUE	No
	1011	False	FALSE	No
	1100	Ordered and Not Equal	FALSE	No
	1101	Ordered and Not Less Than	FALSE	No
	1110	Ordered and Greater Than	FALSE	No
	1111	True	TRUE	No

There are two types of comparisons, ordered and unordered. Ordered comparison operations return TRUE only if both operands are valid numbers and the numbers have the relation specified by the type of comparison and FALSE otherwise. Unordered comparison operations return TRUE if one of the operands is a NaN, or the numbers have the relation specified by the type of comparison; and FALSE otherwise. The “True” and “False” operations return all 1s and all 0s, respectively, regardless of whether any of the source operands is a NaN.

QNaN operands generate an Invalid Operation Exception only if the comparison type is “Less than (or Equal)” and “Greater than (or Equal)”. SNaN operands generate an Invalid Operation (IE) exception for all operations, including “True” and “ False”.

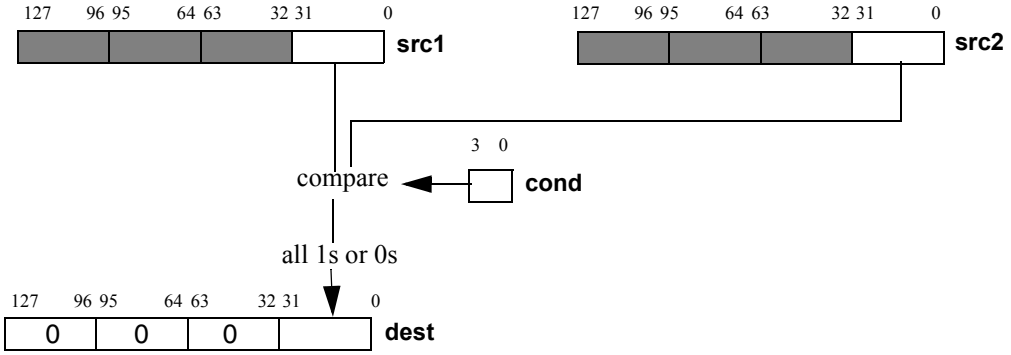
The COMSS instruction requires four operands:

COMSS dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The COMSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
COMSS *xmm1, xmm2, xmm3/mem32, imm8*	0F 25 2E /r /drex0 ib	Compares the low-order single-precision floating-point value in XMM2 register by the low-order single-precision floating-point value in XMM3 register or 32-bit memory location and writes 32 bits of all 1s (TRUE) or all 0s (FALSE) in the low-order doubleword in the destination (XMM1 register).



Related Instructions

COMPS, COMPD, COMSD, CMPPD, CMPPS, CMPSS, CMPSD, COMISD, COMISS, UCOMISD, UCOMISS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
															M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 on page 18).
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.

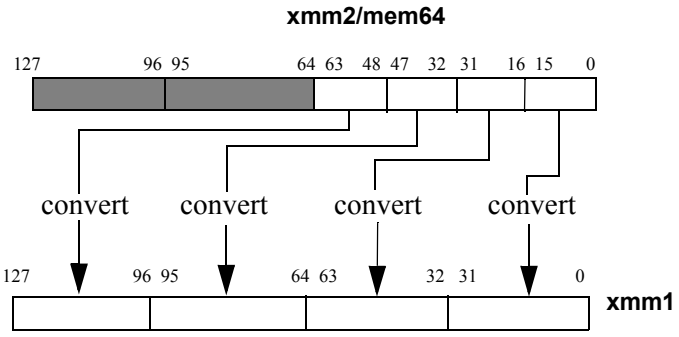
CVTPH2PS Convert 16-Bit Floating-Point to Single-Precision Floating-Point

Converts four packed 16-bit floating-point values in the low-order 64 bits of an XMM register or 64-bit memory location to four packed single-precision floating-point values and writes the converted values in another XMM register. The format of a 16-bit floating-point value is described in Section 1.5, "16-Bit Floating-Point Data Type," on page 7.

If a source value is a denormal, the result is signed zero.

The CVTPH2PS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
CVTPH2PS *xmm1*, *xmm2*/*mem64*	0F 7A 30 /r	Converts four packed 16-bit floating-point values in the low 64 bits of XMM2 or 64-bit memory location to four single-precision floating-point values and writes the results in the destination (XMM1 register).



Related Instructions

CVTPS2PH

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

CVTPS2PH Convert Single-Precision Floating-Point to 16-Bit Floating-Point

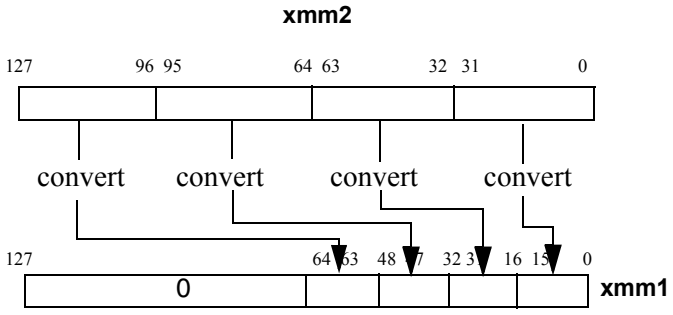
Converts four packed single-precision floating-point values in an XMM register to four packed 16-bit floating-point values and writes the converted values in the low-order 64 bits of another XMM register or to a 64-bit memory location. The high-order 64 bits in the destination register are cleared to 0s. The format of a 16-bit floating-point value is described in Section 1.5, “16-Bit Floating-Point Data Type,” on page 7.

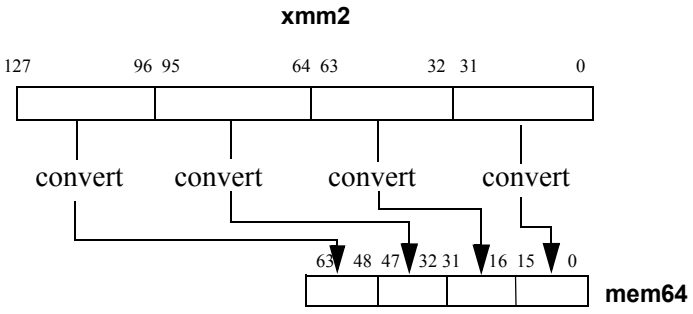
Table 1-10 on page 8 shows the floating-point encodings of supported numbers and non-numbers.

If a source value is smaller than the smallest normalized 16-bit floating-point value, the result is signed zero. If a source value cannot be represented exactly in 16-bit floating-point format, the value is rounded using "truncate" rounding mode.

The CVTPS2PH instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
CVTPS2PH *xmm1/mem64, xmm2*	0F 7A 31 /r	Converts four packed single-precision floating-point values in XMM2 to four 16-bit floating-point values and writes the results in the destination (XMM1 register or memory location).





Related Instructions

CVTPH2PS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

FMADDPD Multiply and Add Packed Double-Precision Floating-Point

Multiplies each of the two packed double-precision floating-point values in first source operand by the corresponding packed double-precision floating-point values in the second source operand, then adds each product to the two corresponding packed double-precision floating-point values in the third source operand. The two results are written to the destination register.

The intermediate products are not rounded; the two infinitely precise products are used in the addition. The results of the addition are rounded, as specified by the rounding mode in MXCSR.

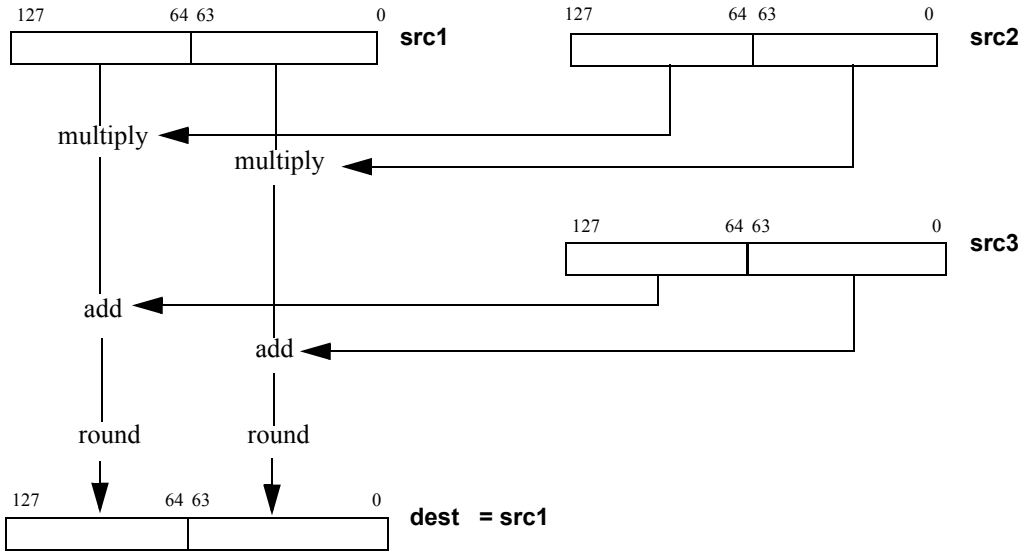
The destination register is an XMM register addressed by the DREX.dest field.

The FMADDPD instruction requires four operands:

FMADDPD dest, src1, src2, src3 *dest = src1* src2 + src3*

The FMADDPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMADDPD *xmm1, xmm1, xmm2, xmm3/mem128*	0F 24 01 /*r* /*drex0*	Multiplies two packed double-precision floating-point values in the second and third operands, then adds the products to the fourth operand and writes the results in the destination (first operand).
FMADDPD *xmm1*, *xmm1*, xmm3/*mem128, xmm2*	0F 24 01 /*r* /*drex1*	
FMADDPD *xmm1*, *xmm2, xmm3/mem128, xmm1*	0F 24 05 /*r* /*drex0*	
FMADDPD *xmm1*, *xmm3/mem128, xmm2, xmm1*	0F 24 05 /*r* /*drex1*	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

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				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+/-zero was multiplied by +/- infinity
	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.

FMADDPS Multiply and Add Packed Single-Precision Floating-Point

Multiplies each of the four single-precision floating-point values in first source operand by the corresponding four single-precision floating-point values in the second source operand, then adds the products to the corresponding four single-precision floating-point values in the third source operand. The four results are written to the destination register.

The intermediate products are not rounded; the four infinitely precise products are used in the addition. The results of the addition are rounded, as specified by the rounding mode in MXCSR.

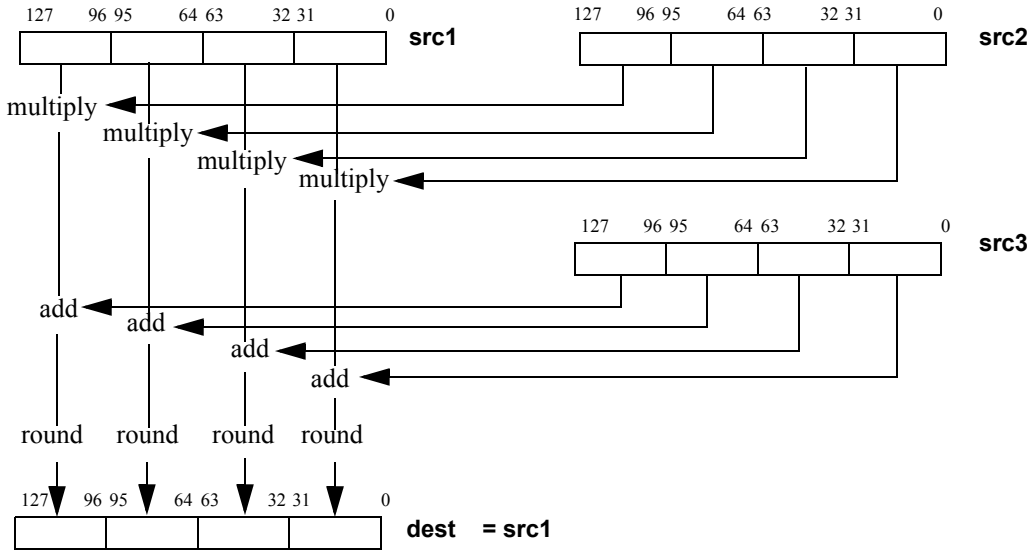
The destination register is an XMM register addressed by the DREX.dest field.

The FMADDPS instruction requires four operands:

FMADDPS dest, src1, src2, src3 *dest = src1* src2 + src3*

The FMADDPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMADDPS *xmm1*, *xmm1, xmm2, xmm3/mem128*	0F 24 00 /r /drex0	Multiplies four packed single-precision floating-point values in the second and third operands, then adds the products to the fourth operand and writes the results in the destination (XMM1 register).
FMADDPS *xmm1*, *xmm1, xmm3/mem128, xmm2*	0F 24 00 /r /drex1	
FMADDPS *xmm1*, *xmm2, xmm3/mem128, xmm1*	0F 24 04 /r /drex0	
FMADDPS *xmm1*, *xmm3/mem128, xmm2, xmm1*	0F 24 04 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

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				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+/-zero was multiplied by +/- infinity
	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.

FMADDSD Multiply and Accumulate Scalar Double-Precision Floating-Point

Multiplies 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, then adds the product to the double-precision floating-point value in the low-order quadword of the third source operand. The low-order quadword result is written to the destination. The high-order quadword of the destination is not modified.

The intermediate product is not rounded; the infinitely precise product is used in the addition. The result of the addition is rounded, as specified by the rounding mode in MXCSR.

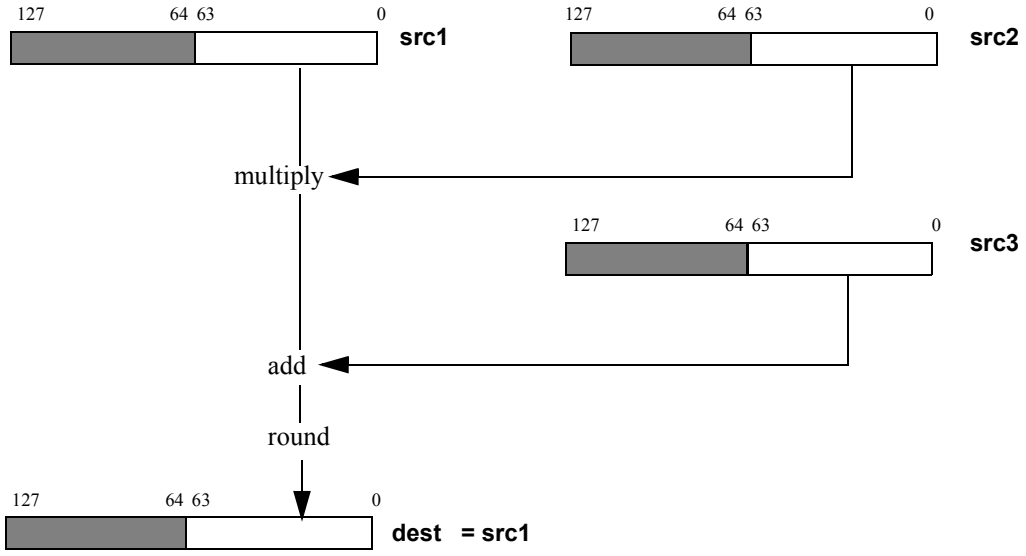
The destination register is an XMM register addressed by the DREX.dest field.

The FMADDSD instruction requires four operands:

FMADDSD dest, src1, src2, src3 *dest = src1* src2 + src3*

The FMADDSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMADDSD *xmm1*, xmm1, *xmm2, xmm3/mem64*	0F 24 03 /r /drex0	Multiplies double-precision floating-point value in the low-order quadword of the second and third operands, then adds the product to the double-precision floating-point value in the low-order quadword of the fourth operand and writes the result in the low order quadword of the destination (XMM1 register).
FMADDSD *xmm1, xmm1, xmm3/mem64, xmm2*	0F 24 03 /r /drex1	
FMADDSD *xmm1*, *xmm2, xmm3/mem64, xmm1*	0F 24 07 /r /drex0	
FMADDSD *xmm1*, *xmm3/mem64, xmm2, xmm1*	0F 24 07 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 multiplied by +/- infinity
	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.

FMADDSS Multiply and Add Scalar Single-Precision Floating-Point

Multiplies the single-precision floating-point value in the low-order doubleword of the first source operand by the low-order single-precision floating-point value in the second source operand, then adds the product to the low-order single-precision floating-point value in the third source operand. The low-order doubleword result is written to the destination. The three high-order doublewords of the destination are not modified.

The intermediate product is not rounded; the infinitely precise product is used in the addition. The result of the addition is rounded, as specified by the rounding mode in MXCSR.

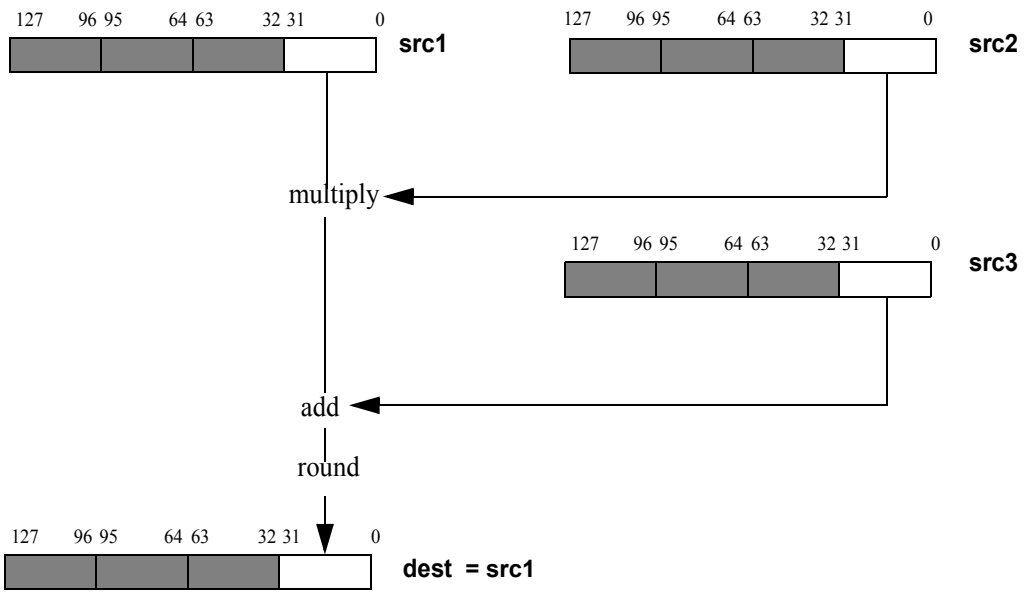
The destination register is an XMM register addressed by the DREX.dest field.

The FMADDSS instruction requires four operands:

FMADDSS dest, src1, src2, src3 *dest = src1* src2 + src3*

The FMADDSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMADDSS *xmm1*, *xmm1, xmm2, xmm3*/*mem32*	0F 24 02 /r /drex0	Multiplies packed single-precision floating-point values in low-order doubleword of the second and third operands, then adds the product to low-order doubleword of the fourth operand and writes the result in the low-order doubleword of the destination (XMM1 register).
FMADDSS *xmm1*, *xmm1, xmm3/mem32, xmm2*	0F 24 02 /r /drex1	
FMADDSS *xmm1*, *xmm2, xmm3*/*mem32, xmm1*	0F 24 06 /r /drex0	
FMADDSS *xmm1*, *xmm3*/*mem32, xmm2, xmm1*	0F 24 06 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 multiplied by +/- infinity
	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.

FMSUBPD Multiply and Subtract 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, then subtracts the corresponding two packed double-precision floating-point values in the third source operand from the products. The two results are written to the destination register.

The intermediate products are not rounded; the two infinitely precise products are used in the subtraction. The results of the subtraction are rounded, as specified by the rounding mode in MXCSR.

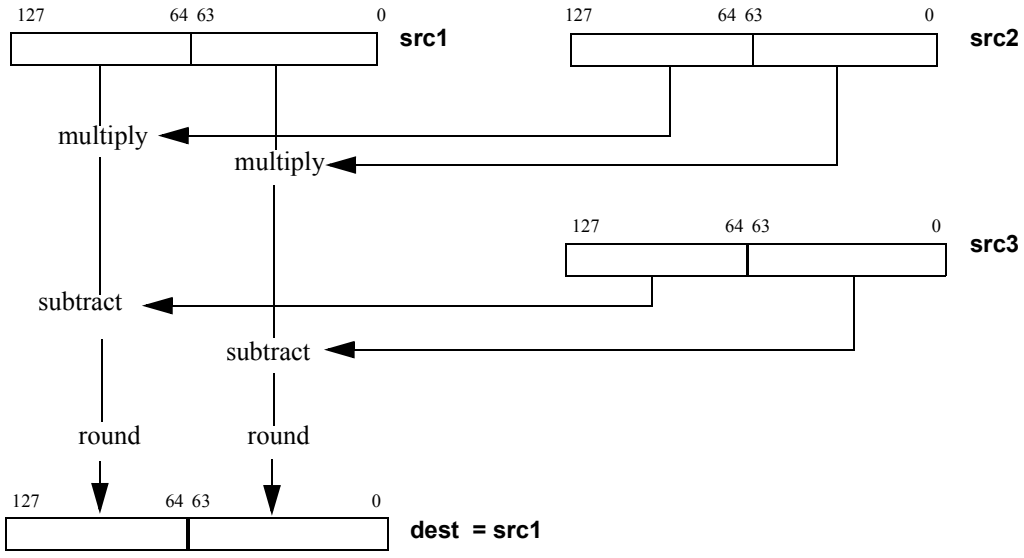
The destination register is an XMM register addressed by the DREX.dest field.

The FMSUBPD instruction requires four operands:

FMSUBPD dest, src1, src2, src3 *dest = src1* src2 - src3*

The FMSUBPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMSUBPD *xmm1*, *xmm1, xmm2, xmm3*/*mem128*	0F 24 09 /r /drex0	Multiplies two packed double-precision floating-point values in the second and third operands, then subtracts the corresponding two packed double-precision floating-point values in the fourth operand from the products and writes the quadword results in the destination (XMM1 register).
FMSUBPD *xmm1*, *xmm1*, xmm3/*mem128, xmm2*	0F 24 09 /r /drex1	
FMSUBPD *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 0D /r /drex0	
FMSUBPD *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 0D /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

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				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+/-zero was multiplied by +/- infinity
	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.

FMSUBPS Multiply and Subtract Packed Single-Precision Floating-Point

Multiplies each of the four packed single-precision floating-point values in the first source operand by the corresponding four packed single-precision floating-point values in the second source operand, then subtracts the corresponding four packed single-precision floating-point values in the third source operand from the products. The four results are written to the destination register.

The intermediate products are not rounded; the four infinitely precise products are used in the subtraction. The results of the subtraction are rounded, as specified by the rounding mode in MXCSR.

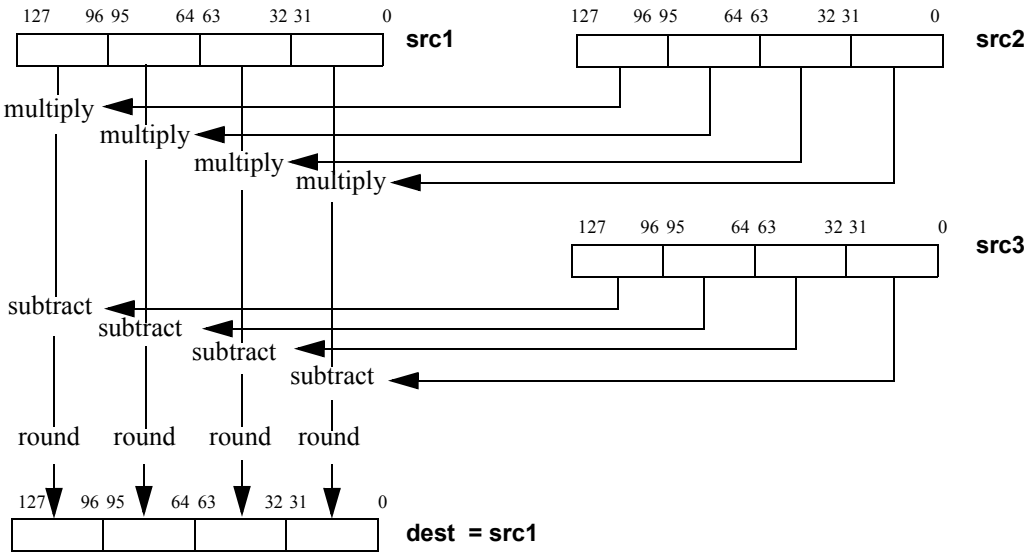
The destination register is an XMM register addressed by the DREX.dest field.

The FMSUBPS instruction requires four operands:

FMSUBPS dest, src1, src2, src3 dest = src1 src2 - src3*

The FMSUBPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMSUBPS *xmm1, xmm1, xmm2, xmm3/mem128*	0F 24 08 /r /drex0	Multiplies four packed single-precision floating-point values in the first and second source operands, then subtracts the corresponding four packed single-precision floating-point values in the third operand from the products and writes the doubleword results in the destination (XMM1 register).
FMSUBPS *xmm1, xmm1, xmm3/mem128, xmm2*	0F 24 08 /r /drex1	
FMSUBPS *xmm1, xmm2, xmm3/mem128, xmm1*	0F 24 0C /r /drex0	
FMSUBPS *xmm1, xmm3/mem128, xmm2, xmm1*	0F 24 0C /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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
	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.

FMSUBSD Multiply and Subtract Scalar Double-Precision Floating-Point

Multiplies 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, then subtracts the double-precision floating-point value in the low-order quadword of the third source operand from the product. The low-order quadword result is written to the destination. The high-order quadword of the destination is not modified.

The intermediate product is not rounded; the infinitely precise product is used in the subtraction. The result of the subtraction is rounded, as specified by the rounding mode in MXCSR.

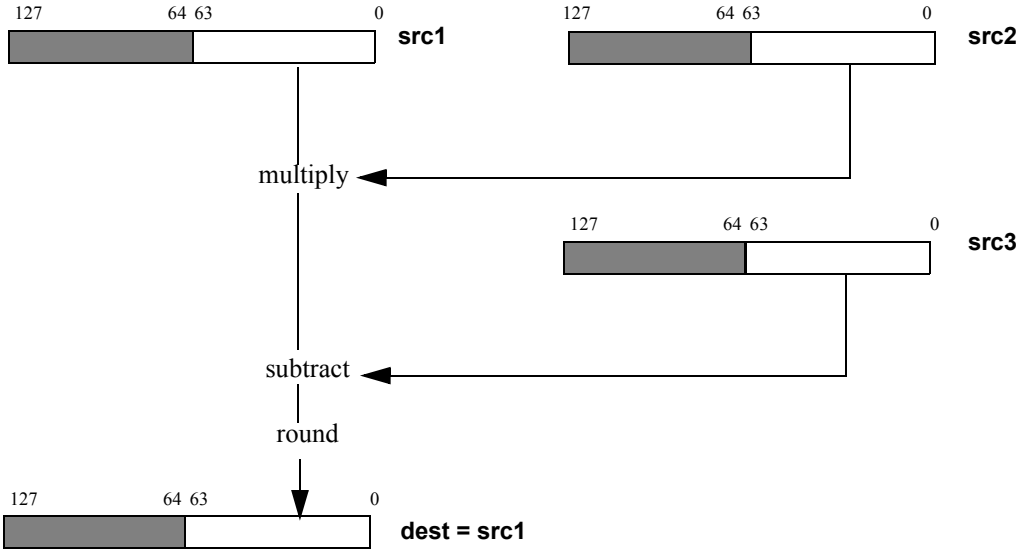
The destination register is an XMM register addressed by the DREX.dest field.

The FMSUBSD instruction requires four operands:

FMSUBSD dest, src1, src2, src3 *dest = src1* src2 - src3*

The FMSUBSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMSUBSD *xmm1*, *xmm1*, *xmm2, xmm3*/*mem64*	0F 24 0B /r /drex0	Multiplies double-precision floating-point value in the low-order quadword of the second and third operands, then subtracts the double-precision floating-point values in the fourth operand from the product and writes the result in the low order quadword of the destination (XMM1 register).
FMSUBSD *xmm1*, *xmm1*, xmm3/*mem64, xmm2*	0F 24 0B /r /drex1	
FMSUBSD *xmm1*, *xmm2, xmm3*/*mem64, xmm1*	0F 24 0F /r /drex0	
FMSUBSD *xmm1*, *xmm3*/*mem64, xmm2, xmm1*	0F 24 0F /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 multiplied by +/- infinity
	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.

FMSUBSS Multiply and Subtract Scalar Single-Precision Floating-Point

Multiplies 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, then subtracts the single-precision floating-point value in the low-order doubleword of the third source operand from the product. The low-order doubleword result is written to the destination. The three high-order doublewords of the destination are not modified.

The intermediate product is not rounded; the infinitely precise product is used in the subtraction. The result of the subtraction is rounded, as specified by the rounding mode in MXCSR.

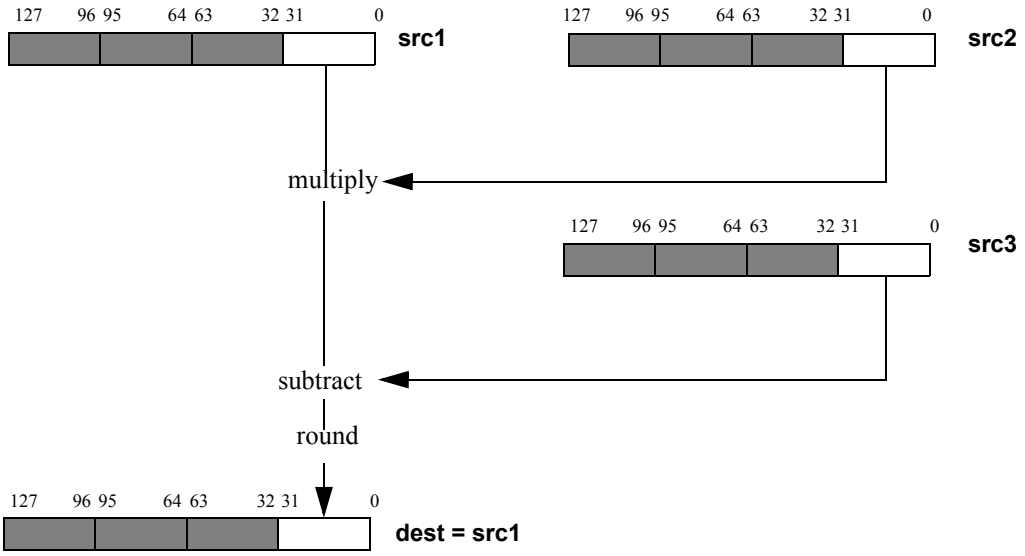
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The FMSUBSS instruction requires four operands:

FMSUBSS dest, src1, src2, src3 *dest = src1* src2 - src3*

The FMSUBSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FMSUBSS *xmm1*, *xmm1*, *xmm2, xmm3*/*mem32*	0F 24 0A /r /drex0	Multiplies single-precision floating-point value in the low-order doubleword of the second and third operands, then subtracts the single-precision floating-point values in the low-order doubleword of the fourth operand from the product and writes the result in the low-order doubleword of the destination (XMM1 register).
FMSUBSS *xmm1*, *xmm1*, xmm3/*mem32, xmm2*	0F 24 0A /r /drex1	
FMSUBSS *xmm1*, *xmm2, xmm3*/*mem32, xmm1*	0F 24 0E /r /drex0	
FMSUBSS *xmm1*, *xmm3*/*mem32, xmm2, xmm1*	0F 24 0E /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDSD, FMADDSS, FNMADDSS, FNMSUBSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 multiplied by +/- infinity
	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.

FNMADDPD Negative Multiply and Add 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, then negates the products and adds them to the corresponding two packed double-precision floating-point values in the third source operand. The two results are written to the destination register.

The intermediate products are not rounded; the two infinitely precise products are negated and then used in the addition. The results of the addition are rounded, as specified by the rounding mode in MXCSR.

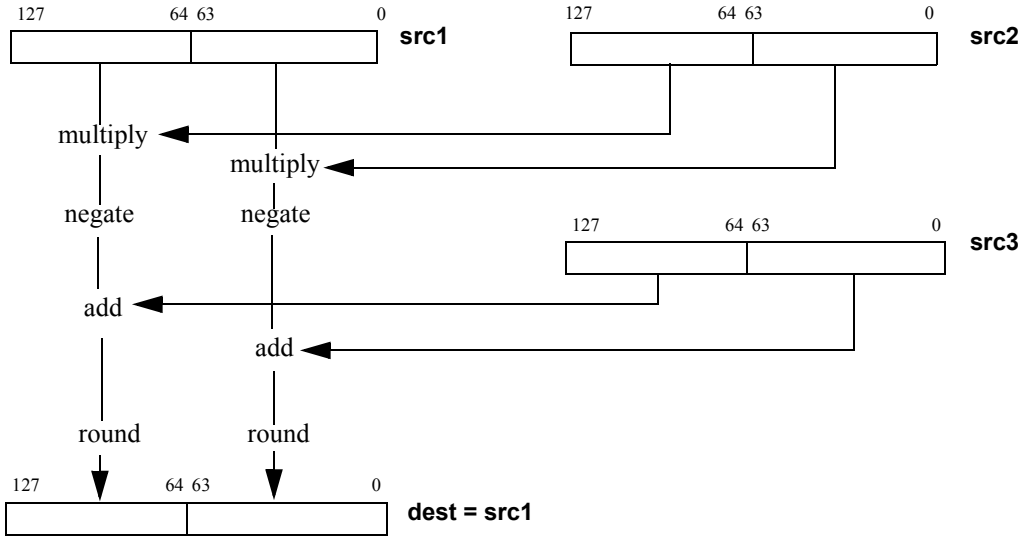
The destination register is an XMM register addressed by the DREX.dest field.

The FNMADDPD instruction requires four operands:

FNMADDPD dest, src1, src2, src3 *dest = - (src1* src2) + src3*

The FNMADDPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMADDPD *xmm1*, *xmm1*, *xmm2, xmm3/mem128*	0F 24 11 /r /drex0	Multiplies two packed double-precision floating-point values in the second and third operands, then negates the products and adds them to the fourth operand and writes the results in the destination (XMM1 register).
FNMADDPD *xmm1*, *xmm1*, xmm3/*mem128, xmm2*	0F 24 11 /r /drex1	
FNMADDPD *xmm1*, *xmm2, xmm3/mem128, xmm1*	0F 24 15 /r /drex0	
FNMADDPD *xmm1*, *xmm3/mem128, xmm2, xmm1*	0F 24 15 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDPD, FMSUBPD, FNMSUBPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

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				
Invalid-operation exception (IE)	X	X	X	A source operand was an SNaN value.
	X	X	X	+/-zero was multiplied by +/- infinity
	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.

FNMADDPS Negative Multiply and Add 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, then negates the products and adds them to the corresponding four packed single-precision floating-point values in the third source operand. The four results are written to the destination register.

The intermediate products are not rounded; the four infinitely precise products are negated and then used in the addition. The results of the addition are rounded, as specified by the rounding mode in MXCSR.

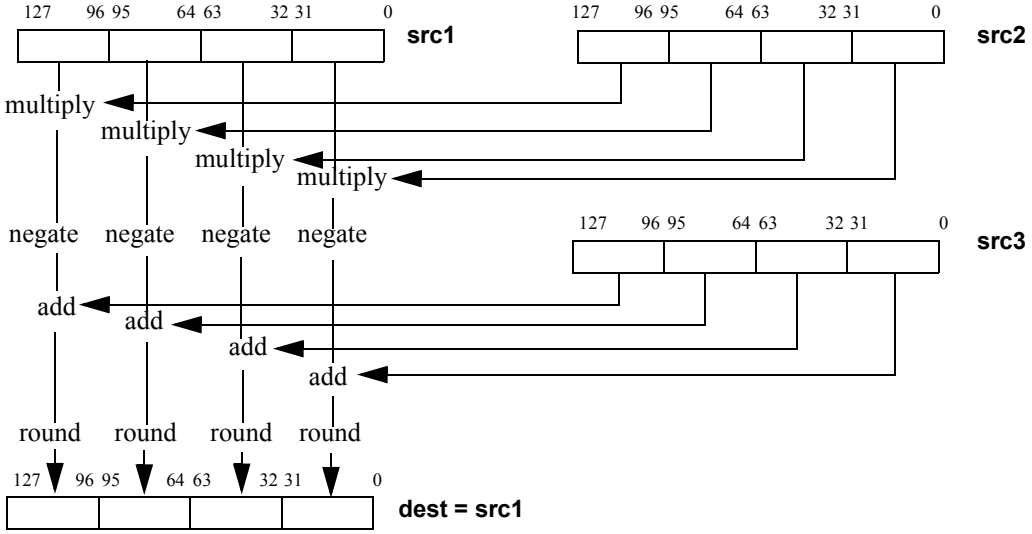
The destination register is an XMM register addressed by the DREX.dest field.

The FNMADDPS instruction requires four operands:

FNMADDPS dest, src1, src2, src3 *dest = -(src1* src2) + src3*

The FNMADDPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMADDPS *xmm1*, *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 10 /r /drex0	Multiplies four packed single-precision floating-point values in the second and third operands, then negates the products and adds them to the fourth operand and writes the results in the destination (XMM1 register).
FNMADDPS *xmm1*, *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 10 /r /drex1	
FNMADDPS *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 14 /r /drex0	
FNMADDPS *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 14 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDPS, FMSUBPS, FNMSUBPS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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
	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.

FNMADDSD Negate Multiply and Add Scalar Double-Precision Floating-Point

Multiplies 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, then negates the product and adds it to the double-precision floating-point value in the low-order quadword of the third source operand. The low-order quadword result is written to the destination. The high-order quadword of the destination is not modified.

The intermediate product is not rounded; the infinitely precise product is negated and then used in the addition. The result of the addition is rounded, as specified by the rounding mode in MXCSR.

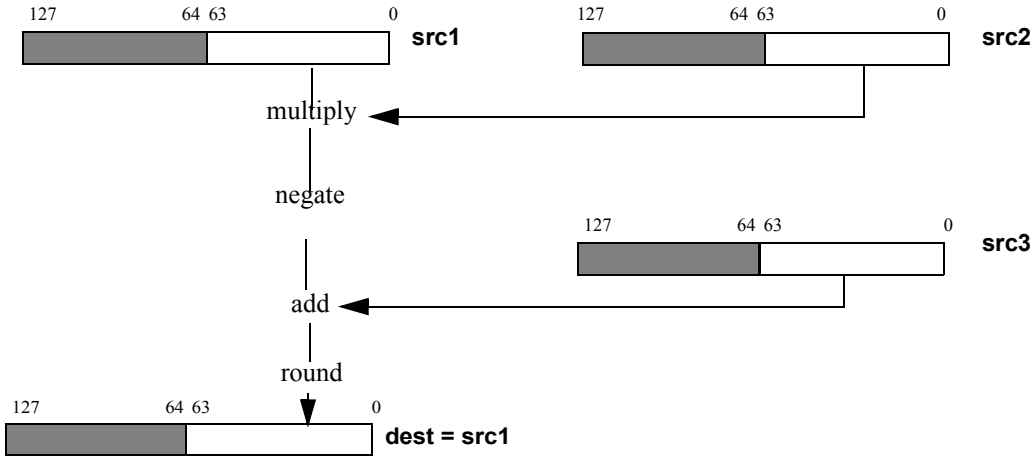
The destination register is an XMM register addressed by the DREX.dest field.

The FNMADDSD instruction requires four operands:

FNMADDSD dest, src1, src2, src3 *dest = - (src1* src2) + src)*

The FNMADDSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMADDSD *xmm1*, *xmm1*, *xmm2, xmm3*/*mem64*	0F 24 13 /r /drex0	Multiplies double-precision floating-point value in the low-order quadword of the second and third operands, then negates the product and adds it to the double-precision floating-point value in the low-order quadword of the fourth operand and writes the result in the low order quadword of the destination (XMM1 register).
FNMADDSD *xmm1*, *xmm1*, xmm3/*mem64, xmm2*	0F 24 13 /r /drex1	
FNMADDSD *xmm1*, *xmm2, xmm3*/*mem64, xmm1*	0F 24 17 /r /drex0	
FNMADDSD *xmm1*, *xmm3*/*mem64, xmm2, xmm1*	0F 24 17 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDSD, FMSUBSD, FNMSUBSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 multiplied by +/- infinity
	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.

FNMADDSS Negative Multiply and Add Scalar Single-Precision Floating-Point

Multiplies 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, then negates the product and adds it to the single-precision floating-point value in the low-order doubleword of the third source operand. The low-order doubleword result is written to the destination. The three high-order doublewords of the destination are not modified.

The intermediate product is not rounded; the infinitely precise product is negated and then used in the addition. The result of the addition is rounded, as specified by the rounding mode in MXCSR.

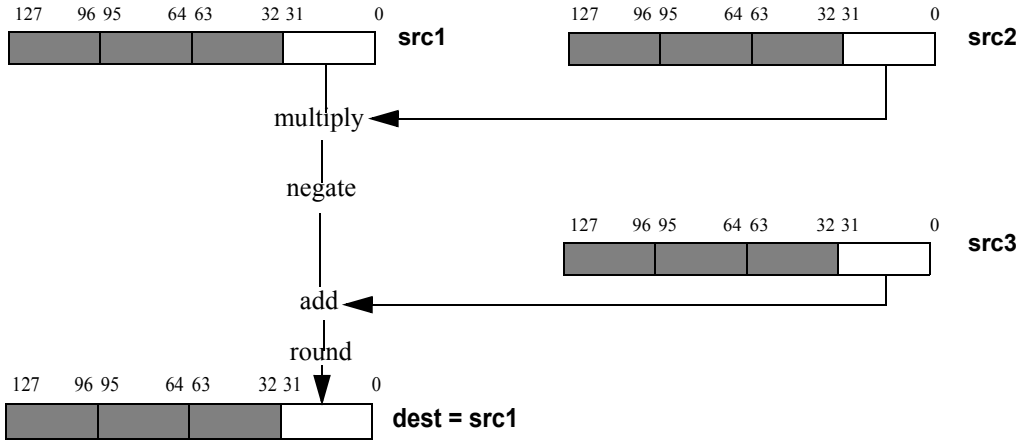
The destination register is an XMM register addressed by the DREX.dest field.

The FNMADDSS instruction requires four operands:

FNMADDSS dest, src1, src2, src3 *dest = - (src1* src2) + src3*

The FNMADDSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMADDSS *xmm1*, *xmm1*, *xmm2, xmm3*/*mem32*	0F 24 12 /r /drex0	Multiplies single-precision floating-point values in low-order doubleword of the second and third operands, then negates the product and adds it to low-order doubleword of fourth operand and writes the result in the low-order doubleword of the destination (XMM1 register).
FNMADDSS *xmm1*, *xmm1*, *xmm3/mem32, xmm2*	0F 24 12 /r /drex1	
FNMADDSS *xmm1*, *xmm2, xmm3*/*mem32, xmm1*	0F 24 16 /r /drex0	
FNMADDSS *xmm1*, *xmm3*/*mem32, xmm2, xmm1*	0F 24 16 /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDSS, FMSUBSS, FNMSUBSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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	+/-zero was multiplied by +/- infinity
	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.

FNMSUBPD Negative Multiply and Subtract 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, then subtracts the corresponding two packed double-precision floating-point values in the third source operand from the negated products.The two results are written to the destination register.

The intermediate products are not rounded; the two infinitely precise products are used in the subtraction. The results of the subtraction are rounded, as specified by the rounding mode in MXCSR.

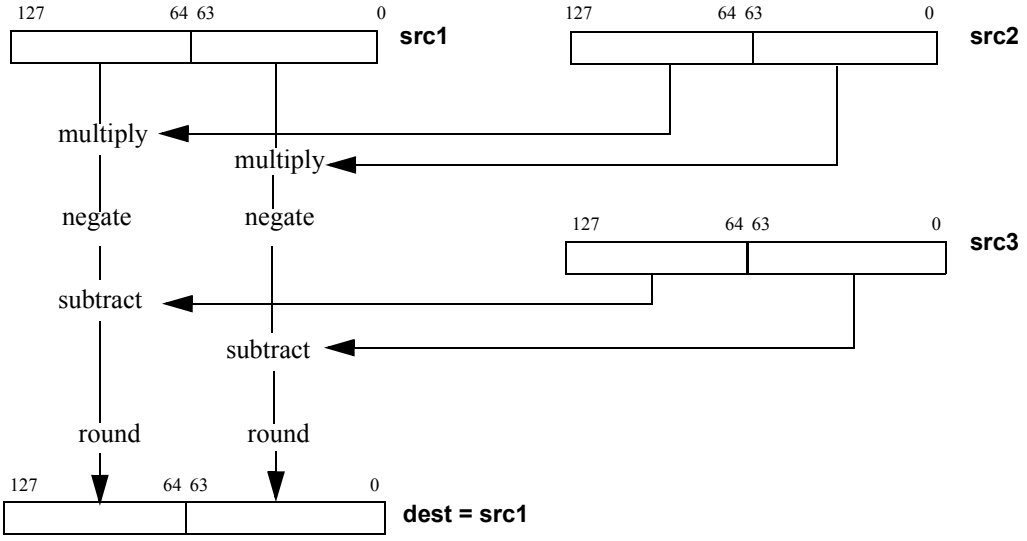
The destination register is an XMM register addressed by the DREX.dest field.

The FNMSUBPD instruction requires four operands:

FNMSUBPD dest, src1, src2, src3 *dest = - (src1* src2) - src3*

The FNMSUBPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMSUBPD *xmm1*, *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 19 /r /drex0	Multiplies two packed double-precision floating-point values in the second and third operands, then subtracts the corresponding two packed double-precision floating-point values in the fourth operand from the negated products and writes the quadword results in the destination (XMM1 register).
FNMSUBPD *xmm1*, *xmm1*, xmm3/*mem128, xmm2*	0F 24 19 /r /drex1	
FNMSUBPD *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 1D /r /drex0	
FNMSUBPD *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 1D /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDPD, FMSUBPD,FNMADDPD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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
	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.

FNMSUBPS Negative Multiply and Subtract Packed Single-Precision Floating-Point

Multiplies 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, then subtracts the corresponding four packed single-precision floating-point values in the third source operand from the negated products. The four results are written to the destination register.

The intermediate products are not rounded; the four infinitely precise products are negated and then used in the subtraction. The results of the subtraction are rounded, as specified by the rounding mode in MXCSR.

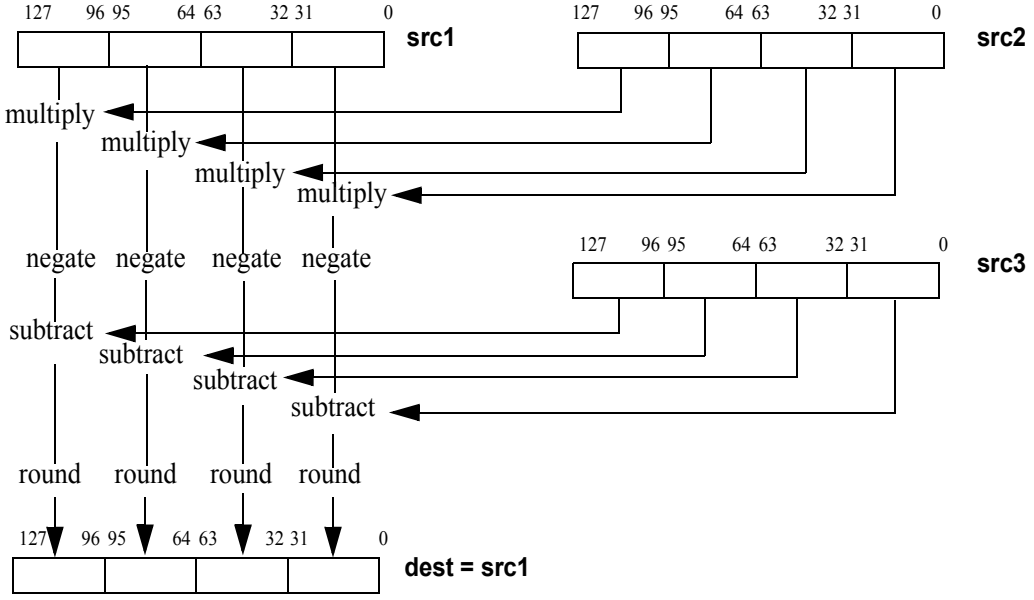
The destination register is an XMM register addressed by the DREX.dest field.

The FNMSUBPS instruction requires four operands:

FNMSUBPS dest, src1, src2, src3 dest = - (src1 src2) - src3*

The FNMSUBPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMSUBPS *xmm1*, *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 18 /r /drex0	Multiplies four packed single-precision floating-point values in the second and third operands, then subtracts the corresponding four packed single-precision floating-point values in the fourth operand from the negated products and writes the doubleword results in the destination (XMM1 register).
FNMSUBPS *xmm1*, *xmm1*, xmm3/*mem128, xmm2*	0F 24 18 /r /drex1	
FNMSUBPS *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 1C /r /drex0	
FNMSUBPS *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 1C /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDPS, FMSUBPS, FNMADDPS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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
	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.

FNMSUBSD Negative Multiply and Subtract Scalar Double-Precision Floating-Point

Multiplies 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, then subtracts the double-precision floating-point value in the low-order quadword of the third source operand from the negated product.The low-order quadword result is written to the destination. The high-order quadword of the destination is not modified.

The intermediate product is not rounded; the infinitely precise product is negated and then used in the subtraction. The result of the subtraction is rounded, as specified by the rounding mode in MXCSR.

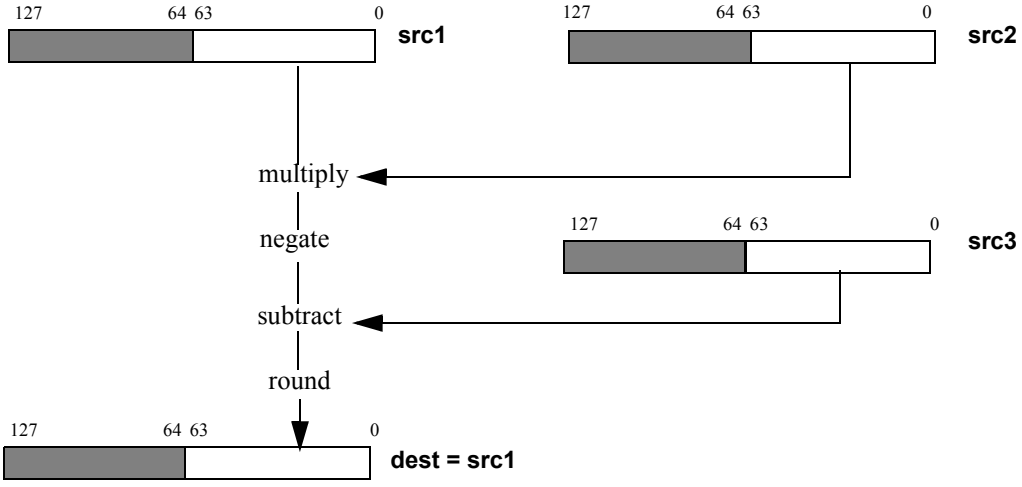
The destination register is an XMM register addressed by the DREX.dest field.

The FNMSUBSD instruction requires four operands:

FNMSUBSD dest, src1, src2, src3 dest = - (src1 src2) - src3*

The FNMSUBSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMSUBSD *xmm1*, *xmm1*, *xmm2, xmm3*/*mem64*	0F 24 1B /r /drex0	Multiplies double-precision floating-point value in the low-order quadword of the second and third operands, then subtracts the double-precision floating-point values in the fourth operand from the negated product and writes the result in the low order quadword of the destination (XMM1 register).
FNMSUBSD *xmm1*, *xmm1*, *xmm3/mem64, xmm2*	0F 24 1B /r /drex1	
FNMSUBSD *xmm1*, *xmm2, xmm3*/*mem64, xmm1*	0F 24 1F /r /drex0	
FNMSUBSD *xmm1*, *xmm3*/*mem64, xmm2, xmm1*	0F 24 1F /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDSD, FMSUBSD, FNMADDSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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	+/-zero was multiplied by +/- infinity
	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.

FNMSUBSS Negative Multiply and Subtract Scalar Single-Precision Floating-Point

Multiplies 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, then subtracts the single-precision floating-point value in the low-order doubleword of the third source operand from the negated product. The low-order doubleword result is written to the destination. The three high-order doublewords of the destination are not modified.

The intermediate product is not rounded; the infinitely precise product is negated and then used in the subtraction. The result of the subtraction is rounded, as specified by the rounding mode in MXCSR.

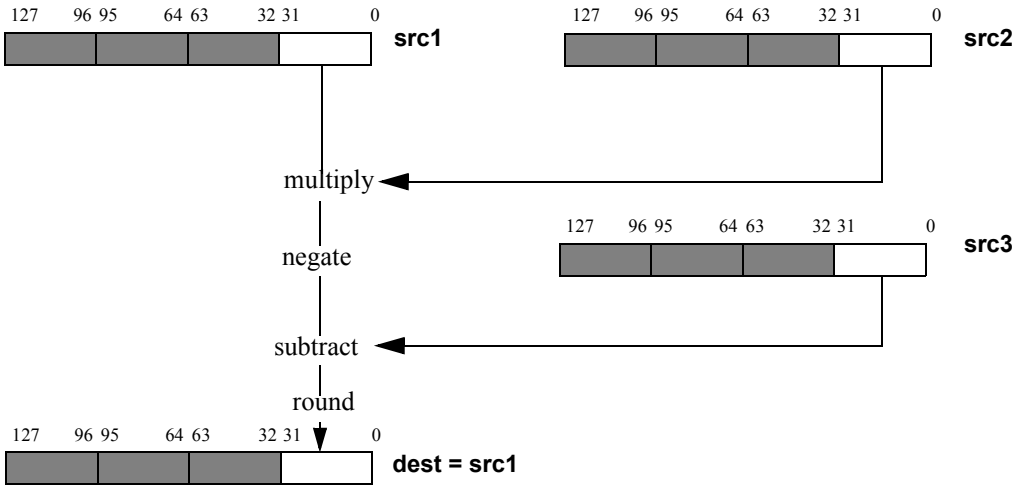
The destination register is an XMM register addressed by the DREX.dest field.

The FNMSUBSS instruction requires four operands:

FNMSUBSS dest, src1, src2, src3 *dest = - (src1* src2) - src3*

The FNMSUBSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FNMSUBSS *xmm1*, *xmm1*, *xmm2, xmm3*/*mem32*	0F 24 1A /r /drex0	Multiplies single-precision floating-point value in the low-order doubleword of the second and third operands, then subtracts the single-precision floating-point values in the low-order doubleword of the fourth operand from the negated product and writes the result in the low-order doubleword of the destination (XMM1 register).
FNMSUBSS *xmm1*, *xmm1*, xmm3/*mem32, xmm2*	0F 24 1A /r /drex1	
FNMSUBSS *xmm1*, *xmm2, xmm3*/*mem32, xmm1*	0F 24 1E /r /drex0	
FNMSUBSS *xmm1*, *xmm3*/*mem32, xmm2, xmm1*	0F 24 1E /r /drex1	



Related Instructions

PMACSDD, PMACSDQH, PMACSDQL, PMACSSDD, PMACSSDQH, PMACSSDQL, PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, FMSUBPD, FNMADDPD, FNMSUBPD FMADDSD, FNMADDSD, FNMSUBSD, FMADDSS, FMSUBSS, FNMADDSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M	M		M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 multiplied by +/- infinity
	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.

FRCZPD Extract Fraction Packed Double-Precision Floating-Point

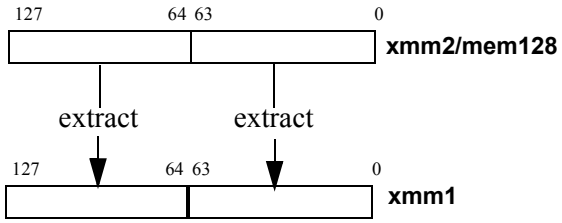
Extracts the fractional portion of each of the two double-precision floating-point values in an XMM register or a 128-bit memory location and writes the result in the corresponding quadword in the destination register. The instruction results are exact.

The rounding mode defined in the MXCSR is ignored.

If the source value is QNaN, it is written to the destination with no exception generated. If the source value is infinity, the instruction returns an indefinite value when the invalid-operation exception (IE) is masked.

The FRCZPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FRCZPD *xmm1*, *xmm2*/*mem128*	0F 7A 11 /r	Extracts the fractional portion of each of two packed double-precision floating-point values in XMM2 register or 128-bit memory location and writes quadword results in the destination (XMM1 register).



Related Instructions

ROUNDPD, ROUNDPS, ROUNDSD, ROUNDSS, FRCZPS, FRCZSS, FRCZSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M			M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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 or infinity
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	The source operand was not an integral value.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

FRCZPS Extract Fraction Packed Single-Precision Floating-Point

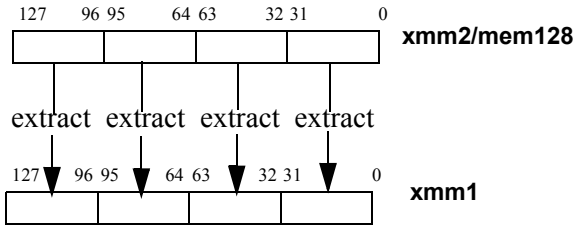
Extracts the fractional portion of each of the four single-precision floating-point values in an XMM register or a 128-bit memory location and writes the result in the corresponding doubleword in the destination register. The instruction results are exact.

The rounding mode indicated in the MXCSR is ignored.

If the source value is QNaN, it is written to the destination with no exception generated. If the source value is infinity, the instruction returns an indefinite value when the invalid-operation exception (IE) is masked.

The FRCZPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FRCZPS *xmm1*, *xmm2*/*mem128*	0F 7A 10 /r	Extracts the fractional portion of each of four packed single-precision floating-point values in XMM2 register or 128-bit memory location and writes corresponding doubleword results in the destination (XMM1 register).



Related Instructions

ROUNDPD, ROUNDPS, ROUNDSD, ROUNDSS, FRCZPD, FRCZSS, FRCZSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M			M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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 or infinity
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	The source operand was not an integral value.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

FRCZSD Extract Fraction Scalar Double-Precision Floating-Point

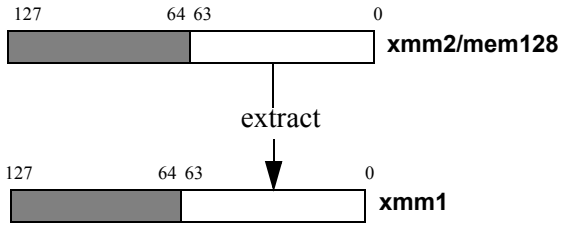
Extracts the fractional portion of the double-precision floating-point value in the low-order quadword of XMM register or 64-bit memory location and writes the result in the low-order quadword in the destination register. The instruction results are exact. The upper double-precision floating-point value in the destination register is not affected.

The rounding mode defined in the MXCSR is ignored.

If the source value is QNaN, it is written to the destination with no exception generated. If the source value is infinity, the instruction returns an indefinite value when the invalid-operation exception (IE) is masked.

The FRCZSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FRCZSD *xmm1*, *xmm2*/*mem64*	0F 7A 13 /r	Extracts the fractional portion of the double-precision floating-point value in the low-order quadword of the XMM2 register or 64-bit memory location and writes the result in the low-order quadword of the destination (XMM1 register).



Related Instructions

ROUNDPD, ROUNDPS, ROUNDSD, ROUNDSS, FRCZPS, FRCZPD, FRCZSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M			M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 or infinity
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	The source operand was not an integral value.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

FRCZSS Extract Fraction Scalar Single-Precision Floating Point

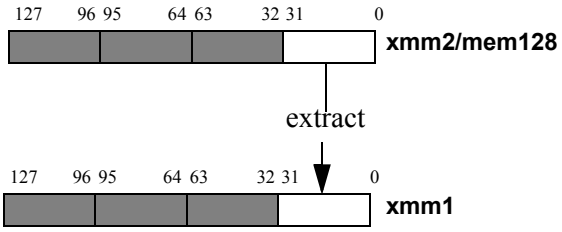
Extracts the fractional portion of the single-precision floating-point value in the low-order doubleword of XMM register or 32-bit memory location and writes the result in the low-order doubleword in the destination register. The instruction results are exact. The upper three single-precision floating-point values in the destination register are not affected.

The rounding mode indicated in the MXCSR is ignored.

If the source value is QNaN, it is written to the destination with no exception generated. If the source value is infinity, the instruction returns an indefinite value when the invalid-operation exception (IE) is masked.

The FRCZSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
FRCZSS *xmm1*, *xmm2*/*mem32*	0F 7A 12 /r	Extracts the fractional portion of the single-precision floating-point value in the low-order doubleword of the XMM2 register or 32-bit memory location and writes the result in the low-order doubleword of the destination (XMM1 register).



Related Instructions

ROUNDPD, ROUNDPS, ROUNDSD, ROUNDSS, FRCZPS, FRCZPD, FRCZSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M	M			M	M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 or infinity
Denormalized-operand exception (DE)	X	X	X	A source operand was a denormal value.
Precision exception (PE)	X	X	X	The source operand was not an integral value.
Underflow exception (UE)	X	X	X	A rounded result was too small to fit into the format of the destination operand.

PCMOV Vector Conditional Moves

Moves bits of either the first source operand or the second source operand into the destination, based on the value of the corresponding bit in a bitwise predicate of the selector operand. If the selector bit is set to 1, the corresponding bit in the first source operand is moved to the destination; otherwise, the corresponding bit from the second source operand is moved to the destination. All moves are 128 bits in length.

This instruction directly implements the C-language ternary "?" operation on each of the 128 bits.

The destination register is an XMM register addressed by the DREX.dest field.

The PCMOV instruction requires four operands:

PCMOV dest, src1, src2, selector

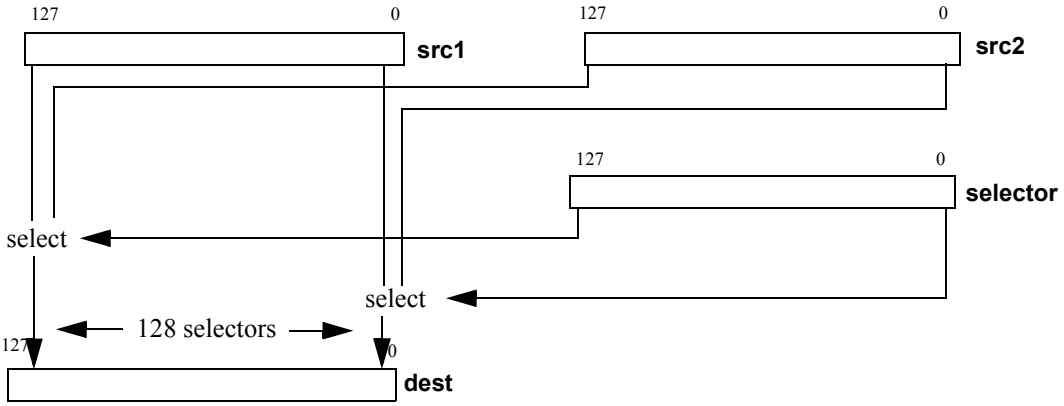
The PCMOV instruction may use instructions to compute the predicate in the selector operand. PCMPEQB (PCMPGTB), PCMPEQW (PCMPGTW) and PCMPEQD (PCMPGTD) compare byte, word and doubleword integer operands, respectively and set the predicate in the destination register to masks of 1s and 0s accordingly. CMPPS (CMPSS) and CMPPD (CMPSD) compare word and doubleword floating-point operands, respectively and provide the predicate for the floating-point instructions.

The PCMOV instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCMOV *xmm1*, *xmm1, xmm2*, *xmm3/mem128*	0F 24 22 /r /drex0	For each bit position of the 128 bit field, moves the bit value from the second source operand to the destination (xmm1 register) when the associated bit in the fourth source operand =1; otherwise, moves bit value from the third source operand to the destination.
PCMOV *xmm1*, *xmm1*, *xmm3/mem128, xmm2*	0F 24 22 /r /drex1	
PCMOV *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 26 /r /drex0	
PCMOV *xmm1*, *xmm3/mem128, xmm2*, *xmm1*	0F 24 26 /r /drex1	

Action

```
for (i=0; i<128; i=++)
   dest[i]:= selector[i] ? source1[i] : source2[i];
```



Related Instructions

FCMOVB, FCMOVBE, FCMOVE, FCMOVNB, FCMOVNBE, FCMOVNE, FCMOVU, FCMOVNU, CMOVcc

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.

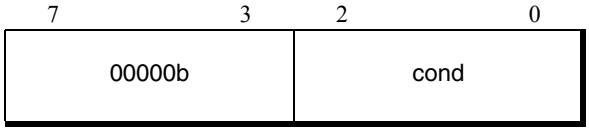
Exception	Real	Virtual 8086	Protected	Cause of Exception
Page fault, #PF		X	X	A page fault resulted from the execution of the instruction.
Alignment Check, #AC		X	X	An unaligned memory reference was performed while alignment checking was enabled while MXCSR.MM=1.

PCOMB Compare Vector 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. The result of each comparison is a 8-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

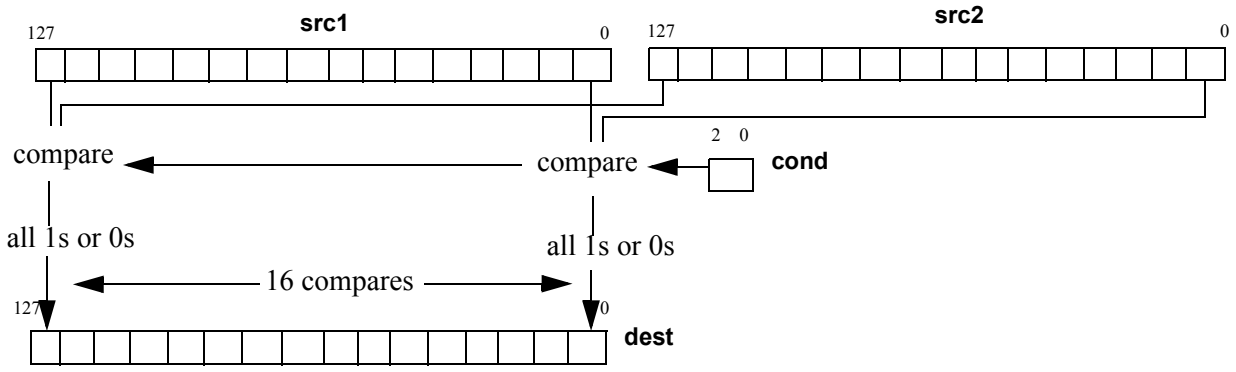
The PCOMB instruction requires four operands:

PCOMB dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMB instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMB *xmm1*, *xmm2, xmm3/mem128, imm8*	0F 25 4C /r /drex0 ib	Compares signed bytes in XMM2 register with corresponding byte in XMM3 register or 128-bit memory location and writes 8 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding byte in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUW, PCOMUD, PCOMUQ, PCOMW, PCOMD, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

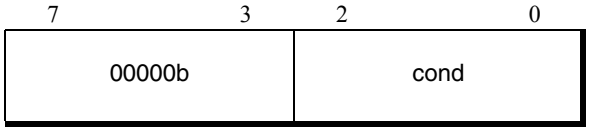
Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMD Compare Vector Signed Doublewords

Compares corresponding packed signed doublewords in the first and second source operands and writes the result of each comparison in the corresponding doubleword of the destination. The result of each comparison is a 32-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

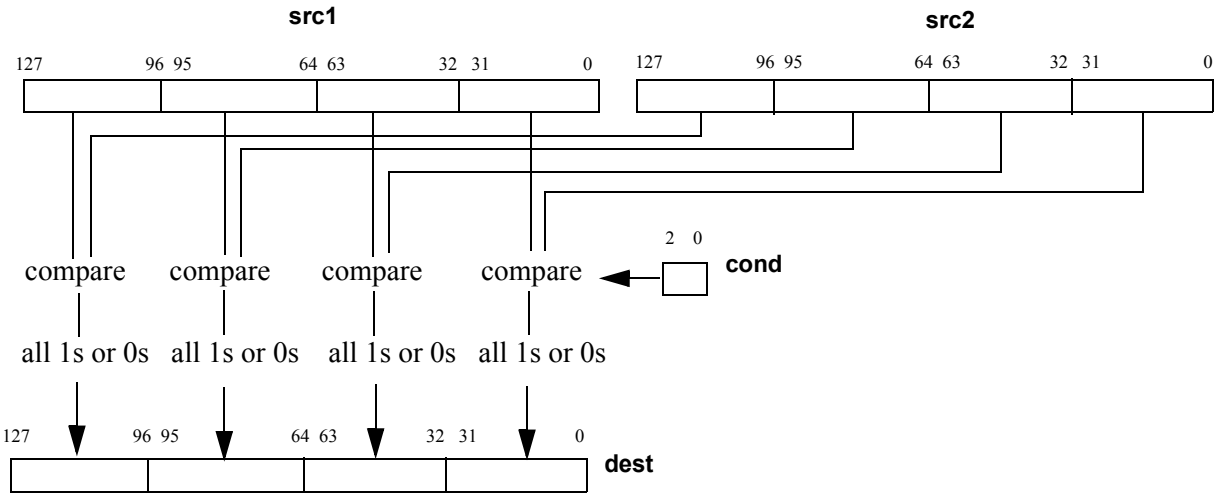
The PCOMD instruction requires four operands:

PCOMD dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMD *xmm1*, *xmm2, xmm3/mem128, imm8*	0F 25 4E /r /drex0 ib	Compares signed doublewords in XMM2 register with corresponding doubleword in XMM3 register or 128-bit memory location and writes 32 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding doubleword in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUW, PCOMUD, PCOMUQ, PCOMB, PCOMW, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

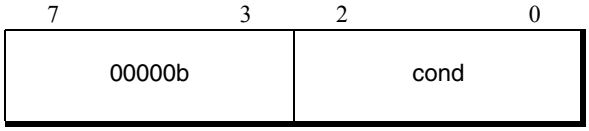
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMQ Compare Vector Signed Quadwords

Compares corresponding packed signed quadwords in the first and second source operands and writes the result of each comparison in the corresponding quadword of the destination. The result of each comparison is a 64-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

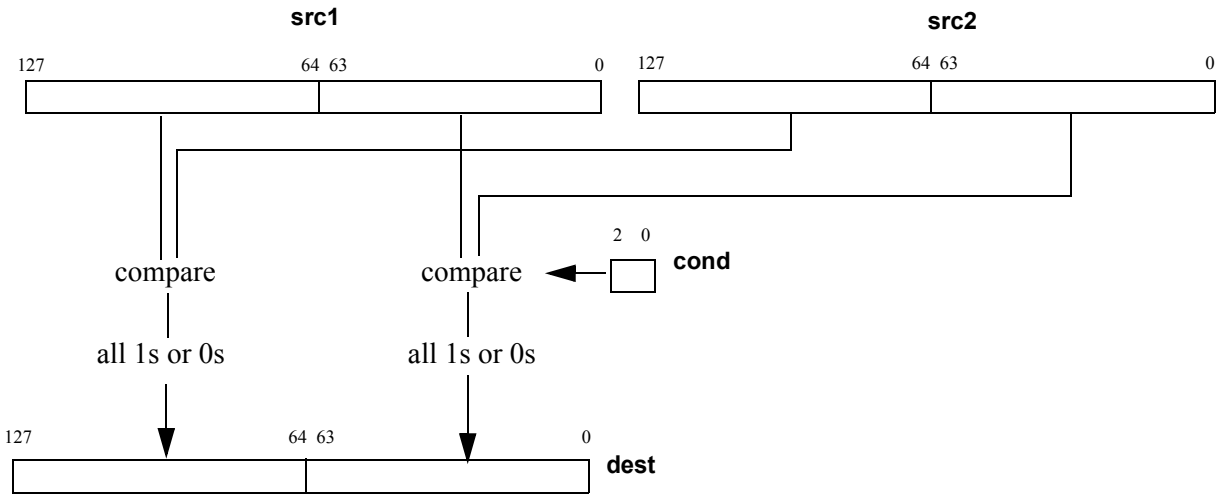
The PCOMQ instruction requires four operands:

PCOMQ dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMQ *xmm1, xmm2, xmm3/mem128, imm8*	0F 25 4F /r /drex0 ib	Compares signed quadwords in XMM2 register with corresponding quadword in XMM3 register or 128-bit memory location and writes 64 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding quadword in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUW, PCOMUD, PCOMUQ, PCOMB, PCOMW, PCOMD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

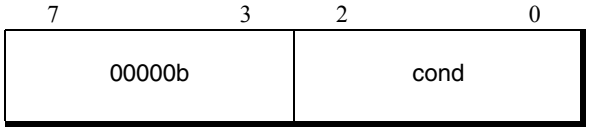
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMUB Compare Vector Unsigned Bytes

Compares corresponding packed unsigned bytes in the first and second source operands and writes the result of each comparison in the corresponding byte of the destination. The result of each comparison is an 8-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

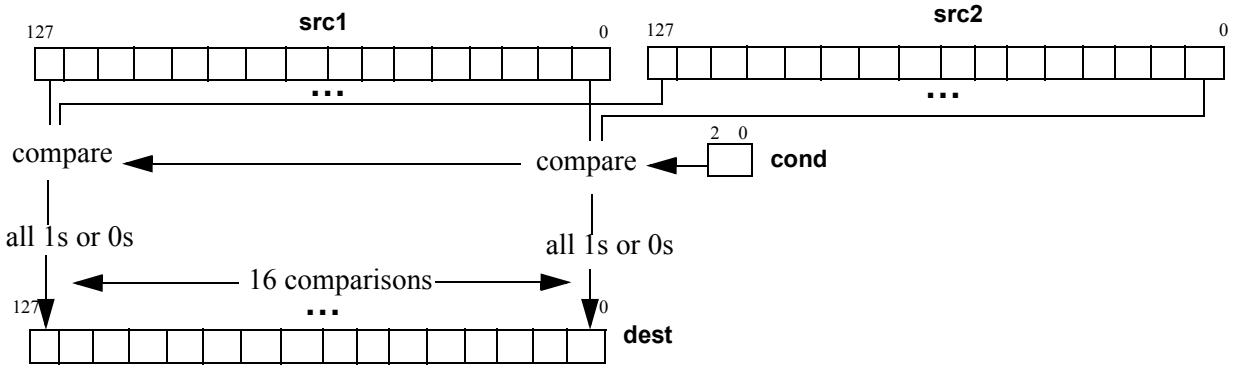
The PCOMUB instruction requires four operands:

PCOMUB dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMUB instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMUB *xmm1*, *xmm2, xmm3*/*mem128, imm8*	0F 25 6C /r /drex0 ib	Compares unsigned bytes in XMM2 register with corresponding byte in XMM3 register or 128-bit memory location and writes 8 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding byte in the destination (XMM1 register).



Related Instructions

PCOMUW, PCOMUD, PCOMUQ, PCOMB, PCOMW, PCOMD, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

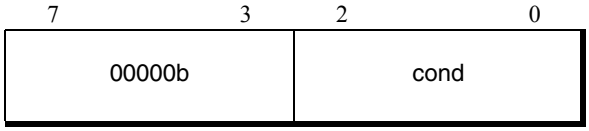
Exception	Real	Virtual 8086	Protected	Cause of Exception
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMUD Compare Vector Unsigned Doublewords

Compares corresponding packed unsigned doublewords in the first and second source operands and writes the result of each comparison in the corresponding doubleword of the destination. The result of each comparison is a 32-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

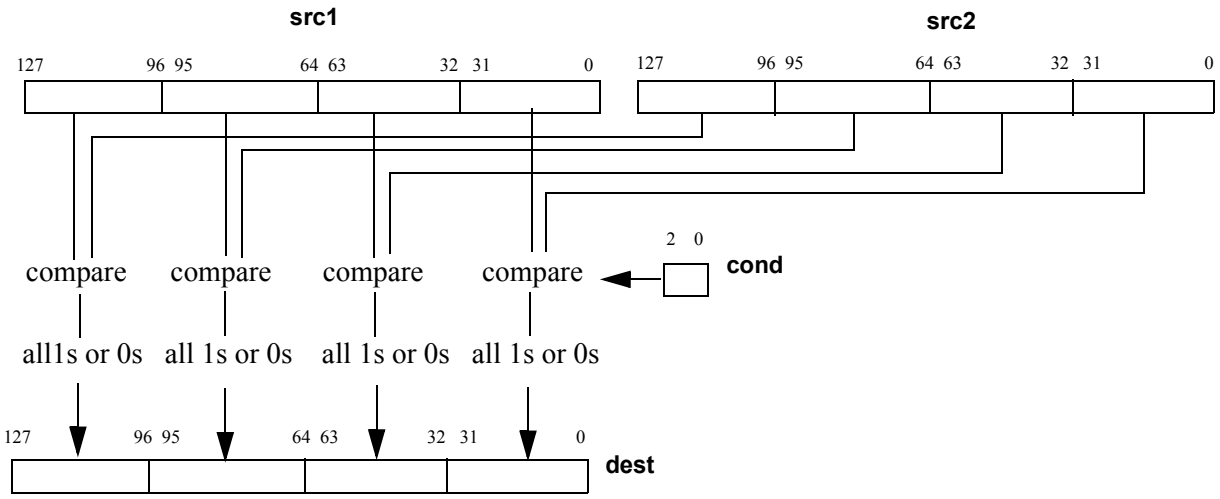
The PCOMUD instruction requires four operands:

PCOMUD dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMUD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMUD *xmm1*, *xmm2, xmm3*/*mem128, imm8*	0F 25 6E /r /drex0 ib	Compares unsigned doublewords in XMM2 register with corresponding doubleword in XMM3 register or 128-bit memory location and writes 32 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding doubleword in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUW, PCOMUQ, PCOMB, PCOMW, PCOMD, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

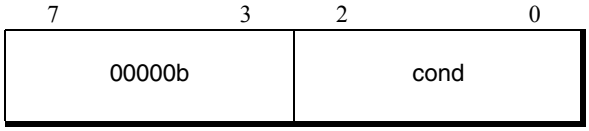
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMUQ Compare Vector Unsigned Quadwords

Compares corresponding packed unsigned quadwords in the first and second source operands and writes the result of each comparison in the corresponding quadword of the destination. The result of each comparison is a 64-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

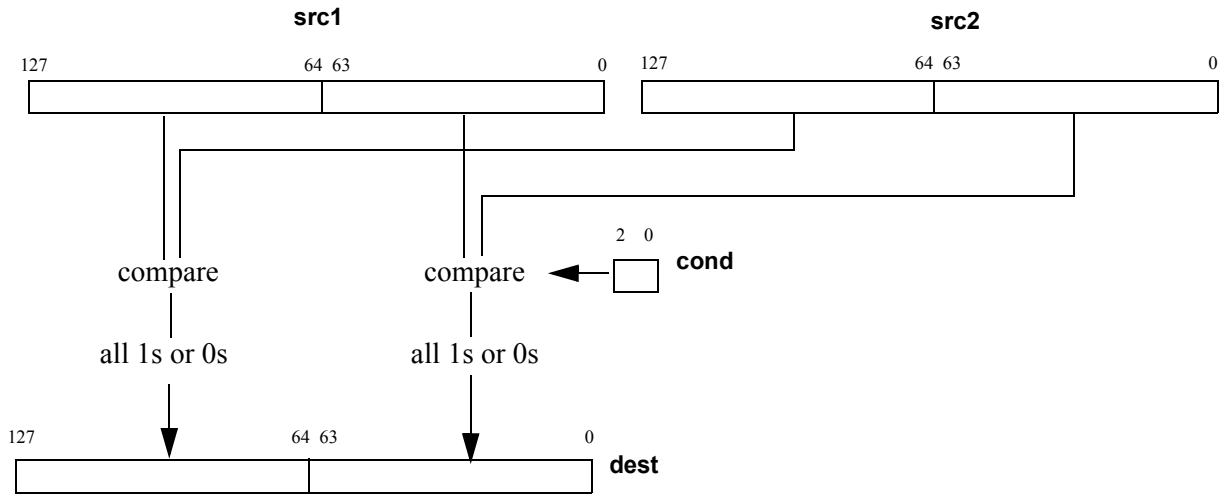
The PCOMUQ instruction requires four operands:

PCOMUQ dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMUQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMUQ *xmm1*, *xmm2, xmm3*/*mem128, imm8*	0F 25 6F /r /drex0 ib	Compares unsigned quadwords in XMM2 register with corresponding quadword in XMM3 register or 128-bit memory location and writes 64 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding quadword in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUW, PCOMUD, PCOMB, PCOMW, PCOMD, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

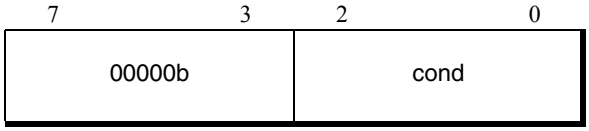
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMUW Compare Vector Unsigned Words

Compares corresponding packed unsigned words in the first and second source operands and writes the result of each comparison in the corresponding word of the destination. The result of each comparison is a 16-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

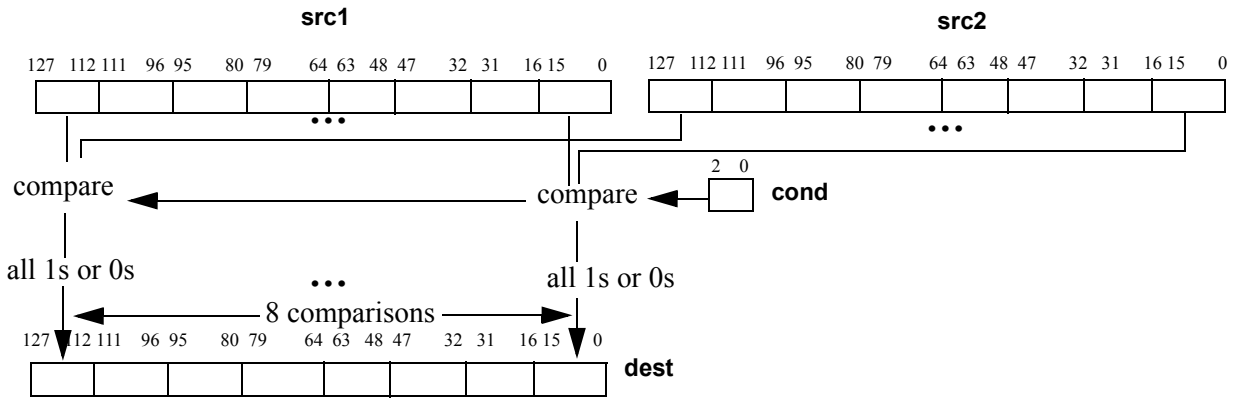
The PCOMUW instruction requires four operands:

PCOMUW dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMUW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMUW *xmm1*, *xmm2, xmm3/mem128, imm8*	0F 25 6D /r /drex0 ib	Compares unsigned words in XMM2 register with corresponding word in XMM3 register or 128-bit memory location and writes 16 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding word in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUD, PCOMUQ, PCOMB, PCOMW, PCOMD, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

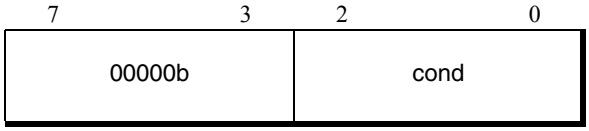
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PCOMW Compare Vector Signed Words

Compares corresponding packed signed words in the first and second source operands and writes the result of each comparison in the corresponding word of the destination. The result of each comparison is a 16-bit value of all 1s (TRUE) or all 0s (FALSE).

The type of comparison is specified by the three low-order bits of the immediate-byte operand, as shown in the following diagram.

PCOM Immediate Operand



Immediate Operand Byte		
Bits	**Descriptions**	
7:3	00000b	
2:0	**cond** – Defines the comparison operation performed on the selected operand.	
	cond	**Comparison Operation**
	000	Less Than
	001	Less Than or Equal
	010	Greater Than
	011	Greater Than or Equal
	100	Equal
	101	Not Equal
	110	False
	111	True

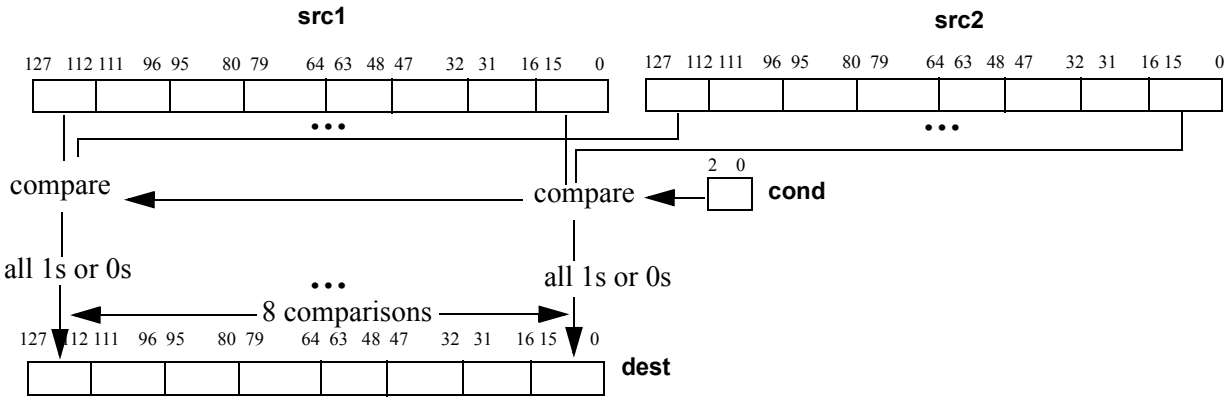
The PCOMW instruction requires four operands:

PCOMW dest, src1, src2, cond

The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PCOMW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PCOMW *xmm1*, *xmm2, xmm3*/*mem128, imm8*	0F 25 4D /r /drex0 ib	Compares signed words in XMM2 register with corresponding word in XMM3 register or 128-bit memory location and writes 16 bits of all 1s (TRUE) or all 0s (FALSE) in the corresponding word in the destination (XMM1 register).



Related Instructions

PCOMUB, PCOMUW, PCOMUD, PCOMUQ, PCOMB, PCOMD, PCOMQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PERMPD Permute Double-Precision Floating-Point

Moves any of the four double-precision values in the source operands to each quadword of the destination XMM register. Each double-precision value of the result can optionally have a logical operation applied to it.

The second source operand (*src2*) is concatenated with the first source operand (*src1*) to form a logical 256-bit source consisting of four double-precision values. The third source operand (*src3*) contains control bytes specifying the source quadword and the logical operation for each destination quadword.

The destination register is an XMM register addressed by the DREX.dest field.

The PERMPD instruction requires four operands:

PERMPD dest, src1, src2, src3

The control bytes for double-precision operands 0 and 1 of the destination are byte 0 and 8, respectively, of the third source.

For each double-precision value of the 16-byte result, the corresponding control byte in *src3* is used as follows:

- bits 1:0 of *src3* select one of the four quadwords from *src2*:*src1*
- bits 7:5 of *src3* select the logical operation applied.

The PERMPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

The control byte is defined in Table 2-1, “PERMPD Control Byte”, on page 120.

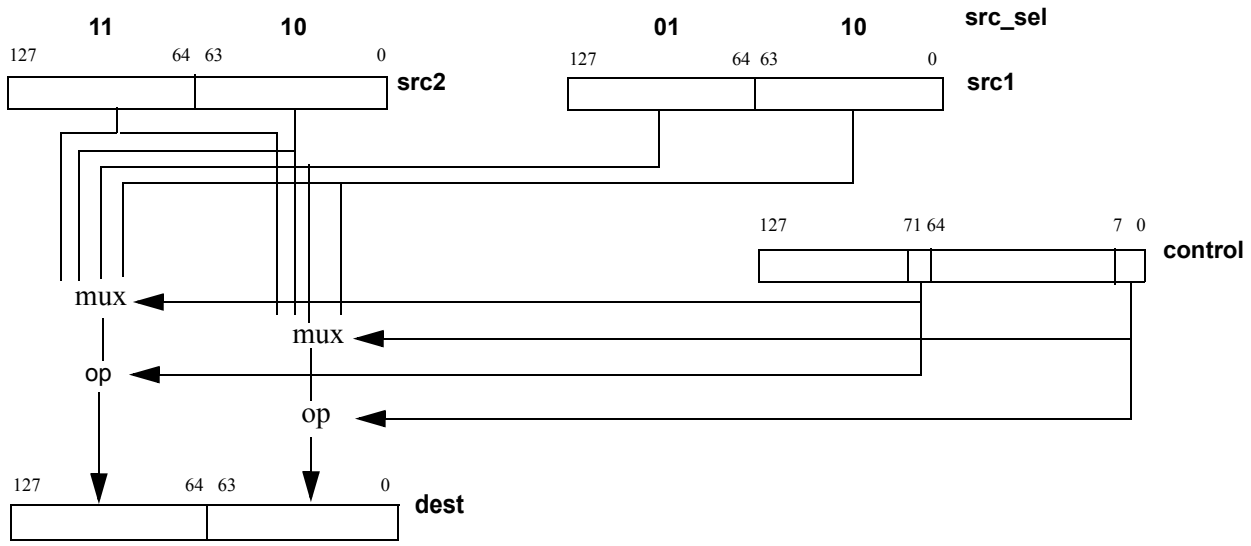
Mnemonic	Opcode	Description
PERMPD *xmm1*, *xmm1, xmm2, xmm3*/*mem128*	0F 24 21 /r /drex0	For each double-precision result, uses corresponding control byte in the fourth operand to perform an operation on one of 4 double-precision operands from the second and third source operands and writes result in destination (xmm1 register).
PERMPD *xmm1*, *xmm1, xmm3*/*mem128, xmm2*	0F 24 21 /r /drex1	
PERMPD *xmm1*, *xmm2*, *xmm3*/*mem128, xmm1*	0F 24 25 /r /drex0	
PERMPD *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 25 /r /drex1	

Table 2-1. PERMPD Control Byte

Control Byte		
Bits	Description	
7:5	**Op -** Defines the logical operation performed on the selected operand.	
	OP	**Operation**
	000	Double-precision source operand
	001	Absolute value of double-precision source operand
	010	Negative value of double-precision source operand
	011	Negative of absolute value of double-precision source operand
	100	0.0
	101	-1.0
	110	1.0
	111	The value of Pi (π), with rounding based on the setting of the rounding control (MXCSR.RC): RC Value 00 0x400921FB54442D18h 01 0x400921FB54442D18h 10 0x400921FB54442D19h 11 0x400921FB54442D18h
4:2	Reserved	
1:0	**Src_Sel -** Selects the double-precision quadword source operand to be operated on.	
	Src_Sel	**Source Selected**
	00	src1[63:0]
	01	src1[127:64]
	10	src2[63:0]
	11	src2[127:64]

Action

```
for (i=0; i<2 i=++)
  dest[i]:= control[i].op (src1|src2)control[i].src_sel;
```



Related Instructions

PSHUFHW, PSHUFD, PSHUFLW, PSHUFW, PPERM, PERMPS

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PERMPS Permute and Modify Single-Precision Floating Point

Moves any of the eight single-precision values in the source operands to each doubleword of the destination XMM register. Each single-precision value of the result can optionally have a logical operation applied to it.

The second source operand (*src2*) is concatenated with the first source operand (*src1*) to form a logical 256-bit source consisting of eight single-precision values. The third source operand (*src3*) contains control bytes specifying the source doubleword and the logical operation for each destination doubleword.

The destination register is an XMM register addressed by the DREX.dest field.

The PERMPS instruction requires four operands:

PERMPS dest, src1, src2, src3

The control bytes for single-precision operands 0, 1, 2 and 3 of the destination are byte 0, 4, 8 and 12, respectively, of the third source.

For each single-precision value of the 16-byte result, the corresponding control byte in src3 is used as follows:

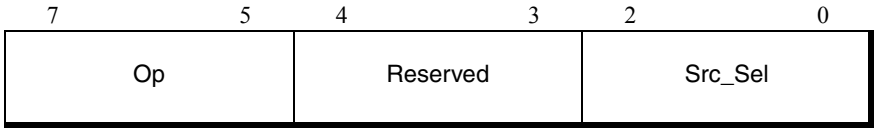
- bits 2:0 of src3 select one of the 8 doublewords from src2:src1
- bits 7:5 of src3 select the operation applied.

The PERMPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

The control byte is defined in Table 2-2, “PERMPS Control Byte”, on page 124.

Mnemonic	Opcode	Description
PERMPS *xmm1*, *xmm1, xmm2, xmm3*/*mem128*	0F 24 20 /r /drex0	For each single-precision result, uses corresponding control byte in the fourth operand to perform an operation on one of 8 single-precision operands from the second and third source operands and writes result in destination (xmm1 register).
PERMPS *xmm1*, *xmm1, xmm3*/*mem128, xmm2*	0F 24 20 /r /drex1	
PERMPS *xmm1*, *xmm2*, *xmm3*/*mem128, xmm1*	0F 24 24 /r /drex0	
PERMPS *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 24 /r /drex1	

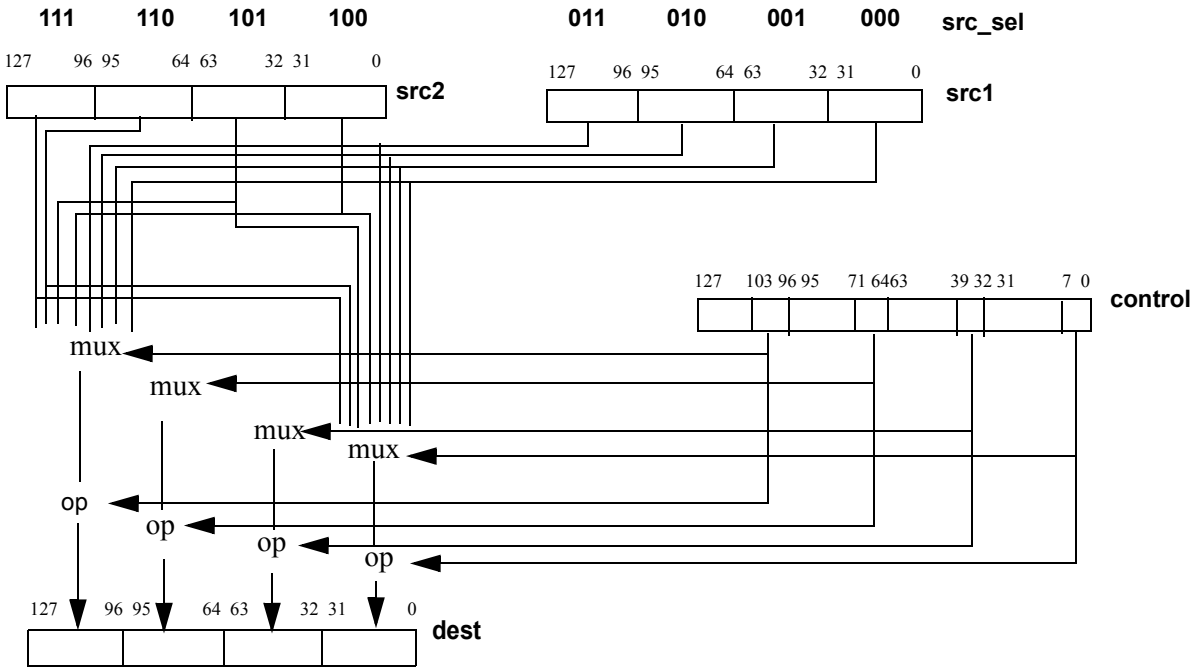
Table 2-2. PERMPS Control Byte



Control Byte		
Bits	Description	
7:5	**Op -** Defines the operation performed on the selected operand.	
	OP	**Operation**
	000	Single-precision source operand
	001	Absolute value of single-precision source operand
	010	Negative value of single-precision source operand
	011	Negative of absolute value of single-precision source operand
	100	+0.0
	101	-1.0
	110	+1.0
	111	The value of Pi (π), with rounding based on the setting of the rounding control (MXCSR.RC): RC Value 00 0x40490FDBh 01 0x40490FDAh 10 0x40490FDBh 11 0x40490FDAh
4:3	Reserved	
2:0	**Src_Sel** - Selects the single-precision doubleword source operand to be operated on.	
	Src_Sel	**Source Selected**
	000	src1[31:0]
	001	src1[63:32]
	010	src1[95:64]
	011	src1[127:96]
	100	src2[31:0]
	101	src2[63:32]
	110	src2[95:64]
	111	src2[127:96]

Action

```
for (i=0; i<4 i=++)
  dest[i]:= control[i].op (src1|src2)control[i].src_sel;
```



Related Instructions

PSHUFHW, PSHUFD, PSHUFLW, PSHUFW, PPERM, PERMPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

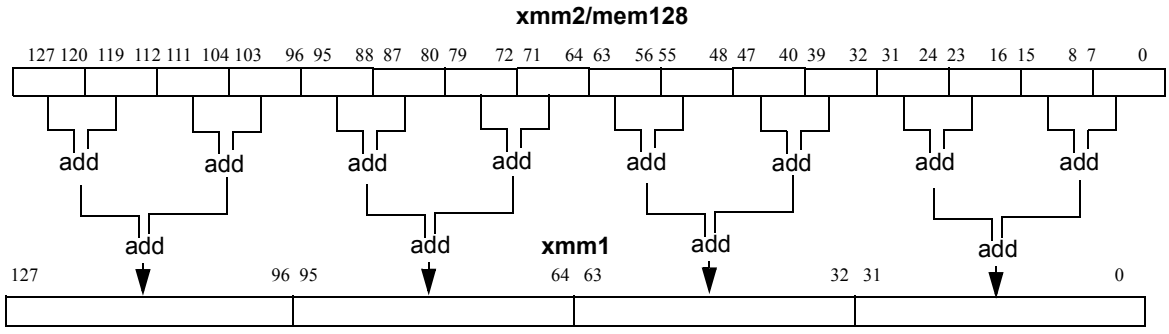
Exception	Real	Virtual 8086	Protected	Cause of Exception
Device not available, #NM	X	X	X	The task-switch bit (TS) of CR0 was set to 1.
Stack, #SS	X	X	X	A memory address exceeded the stack segment limit or was non-canonical.
General protection, #GP	X	X	X	A memory address exceeded a data segment limit or was non-canonical.
			X	A null data segment was used to reference memory.
	X	X	X	The memory operand was not aligned on a 16-byte boundary while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDBD Packed Horizontal Add Signed Byte to Signed Doubleword

Adds four successive 8-bit signed integer values from the second source operand and packs the sign-extended results of the additions in a doubleword 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 PHADDBD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDBD *xmm1*, *xmm2/mem128*	0F 7A 42 /r	Adds four successive 8-bit signed integer values in an XMM register or 128-bit memory location and packs the 32-bit results in the destination XMM register.



Related Instructions

PHADDBW, PHADDBQ, PHADDWD, PHADDWQ, PHADDDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

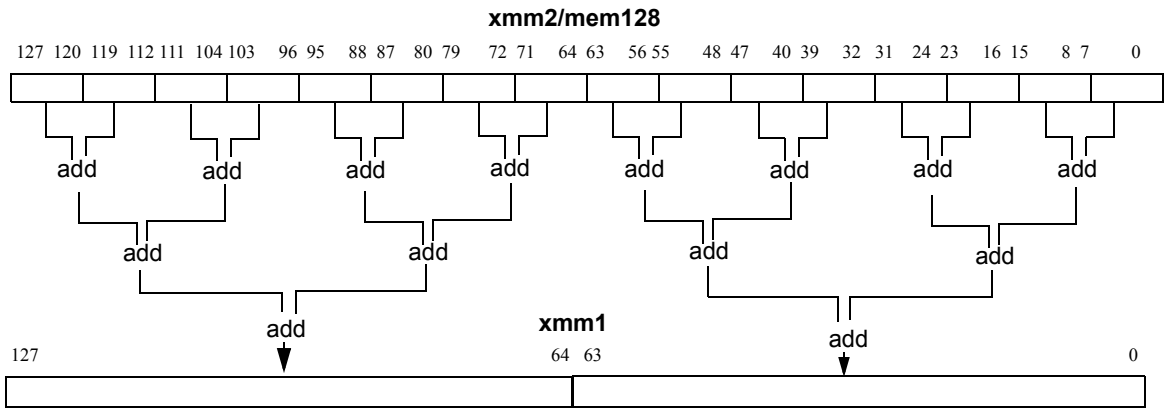
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDBQ Packed Horizontal Add Signed Byte to Signed Quadword

Adds eight successive 8-bit signed integer values from the second source operand and packs the sign-extended results of the additions in a quadword 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 PHADDBQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDBQ *xmm1*, *xmm2*/*mem128*	0F 7A 43 /r	Adds eight successive 8-bit signed integer values in an XMM register or 128-bit memory location and packs the 32-bit results in the destination XMM register.



Related Instructions

PHADDBW, PHADDBD, PHADDWD, PHADDWQ, PHADDDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

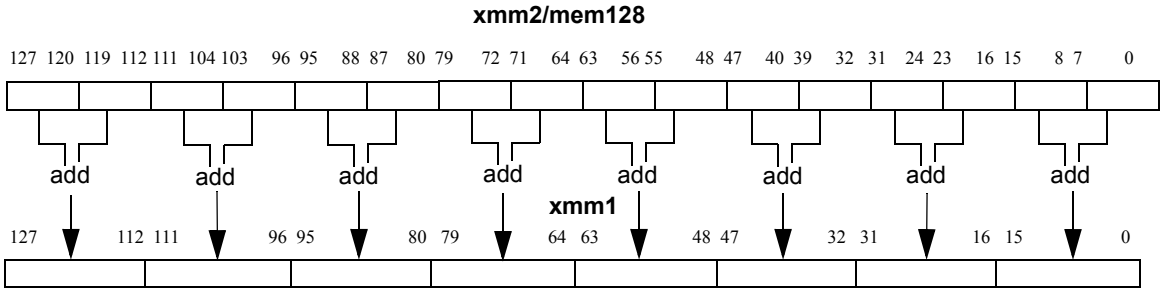
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDBW Packed Horizontal Add Signed Byte to Signed Word

Adds each adjacent pair of 8-bit signed integer values from the second source operand and packs the sign-extended 16-bit integer result of each addition 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 PHADDBW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDBW *xmm1*, *xmm2*/*mem128*	0F 7A 41 /r	Adds each adjacent pair of 8-bit signed integer values in an XMM register or 128-bit memory location and packs the 16-bit results in the destination XMM register.



Related Instructions

PHADDBD, PHADDBQ, PHADDWD, PHADDWQ, PHADDDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

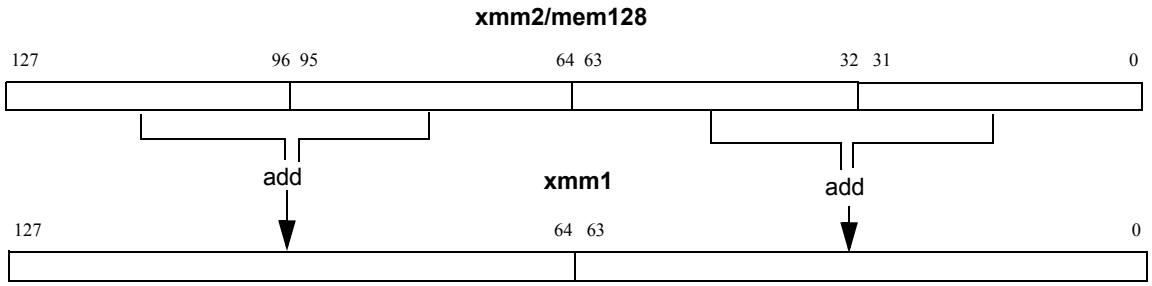
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDDQ Packed Horizontal Add Signed Doubleword to Signed Quadword

Adds each adjacent pair of 32-bit signed integer values from the second source operand and packs the sign-extended results of the additions in a quadword 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 PHADDDQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDDQ *xmm1*, *xmm2*/*mem128*	0F 7A 4B /r	Adds each adjacent pair of 32-bit signed integer values in an XMM register or 128-bit memory location and packs the 64-bit results in the destination XMM register.



Related Instructions

PHADDBW, PHADDBD, PHADDBQ, PHADDWD, PHADDWQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

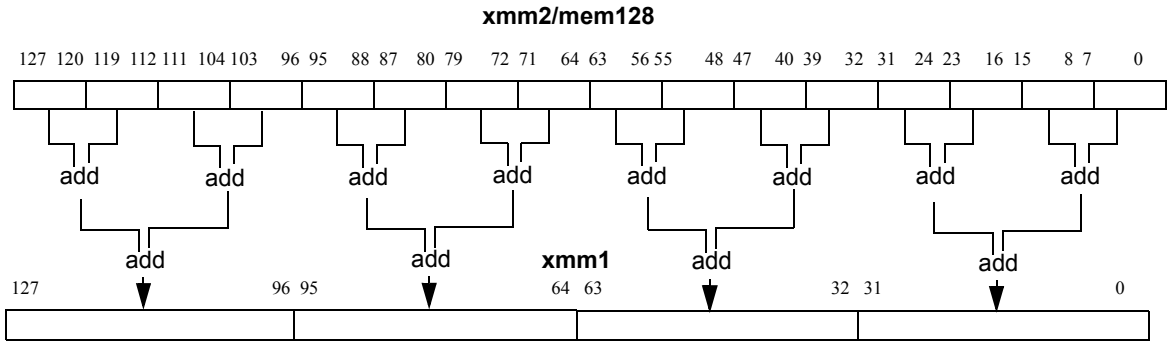
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDUBD Packed Horizontal Add Unsigned Byte to Doubleword

Adds four successive 8-bit unsigned integer values from the second source operand and packs the results of the additions in a doubleword 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 PHADDUBD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDUBD *xmm1*, *xmm2*/*mem128*	0F 7A 52 /r	Adds four successive 8-bit unsigned integer values in an XMM register or 128-bit memory location and packs the 32-bit results in the destination XMM register.



Related Instructions

PHADDUBW, PHADDUBQ, PHADDUWD, PHADDUWQ, PHADDUDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

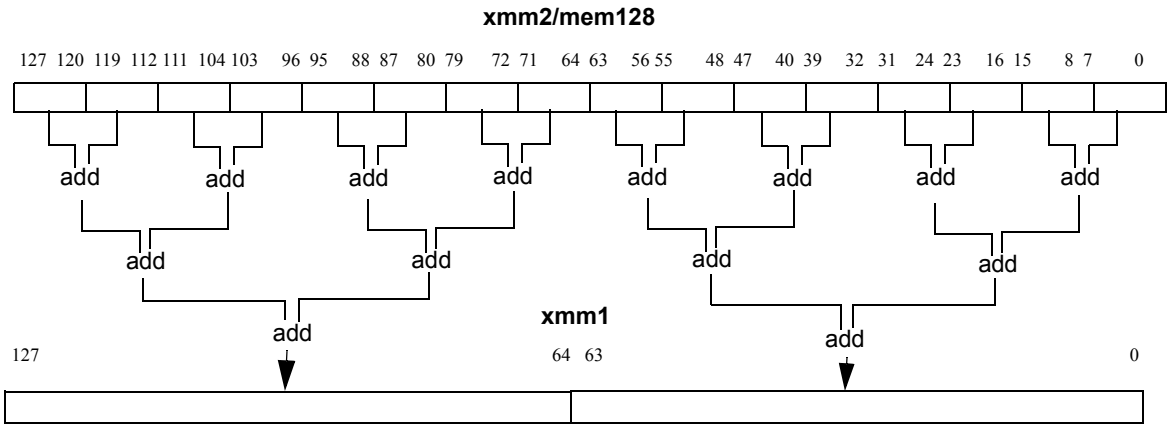
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDUBQ Packed Horizontal Add Unsigned Byte to Quadword

Adds eight successive 8-bit unsigned integer values from the second source operand and packs the results of the additions in a quadword 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 PHADDUBQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDUBQ *xmm1*, *xmm2*/*mem128*	0F 7A 53 /r	Adds eight successive 8-bit unsigned integer values in an XMM register or 128-bit memory location and packs the 64-bit results in the destination XMM register.



Related Instructions

PHADDUBW, PHADDUBD, PHADDUWD, PHADDUWQ, PHADDUDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

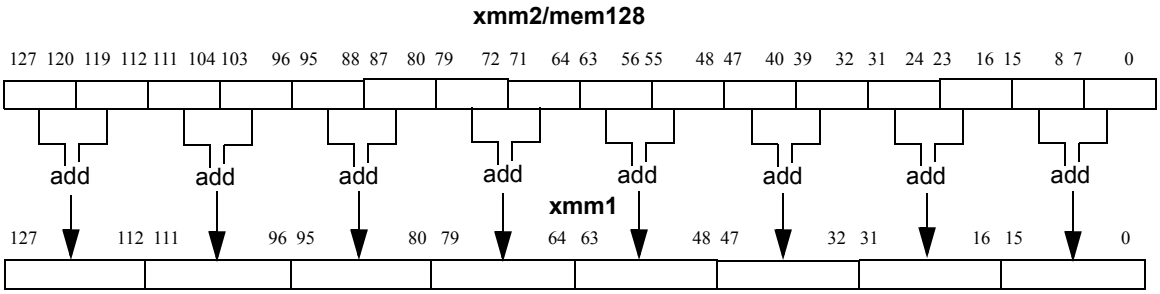
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDUBW Packed Horizontal Add Unsigned Byte to Word

Adds each adjacent pair of 8-bit unsigned integer values from the second source operand and packs the 16-bit integer result of each addition 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 PHADDUBW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDUBW *xmm1*, *xmm2/mem128*	0F 7A 51 /r	Adds each adjacent pair of 8-bit unsigned integer values in an XMM register or 128-bit memory location and packs the 16-bit results in the destination XMM register.



Related Instructions

PHADDUBD, PHADDUBQ, PHADDUWD, PHADDUWQ, PHADDUDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

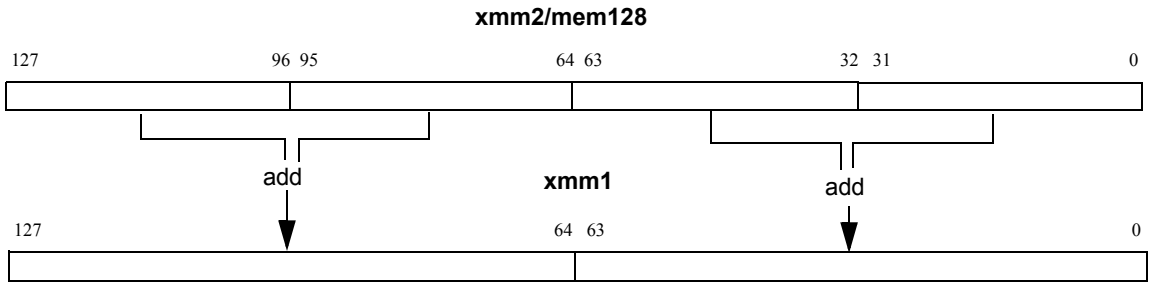
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDUDQ Packed Horizontal Add Unsigned Doubleword to Quadword

Adds each adjacent pair of 32-bit unsigned integer values from the second source operand and packs the results of the additions in a quadword 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 PHADDUDQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDUDQ *xmm1*, *xmm2*/*mem128*	0F 7A 5B /r	Adds each adjacent pair of 32-bit unsigned integer values in an XMM register or 128-bit memory location and packs the 64-bit results in the destination XMM register.



Related Instructions

PHADDUBW, PHADDUBD, PHADDUBQ, PHADDUWD, PHADDUWQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

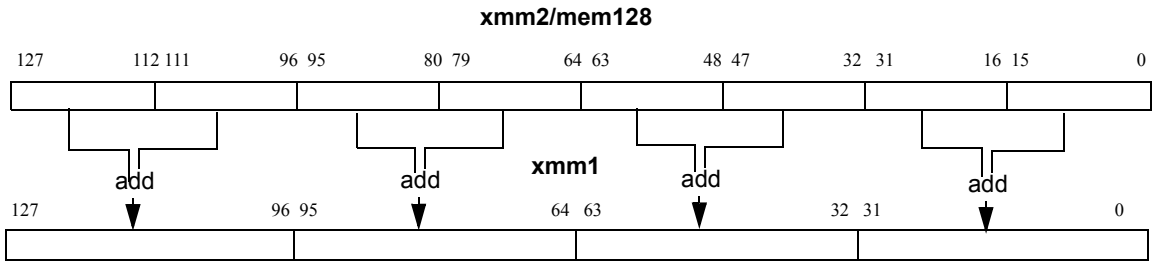
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDUWD Packed Horizontal Add Unsigned Word to Doubleword

Adds each adjacent pair of 16-bit unsigned integer values from the second source operand and packs the results of the addition in a doubleword 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 PHADDUWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDUWD *xmm1*, *xmm2*/*mem128*	0F 7A 56 /r	Adds each adjacent pair of 16-bit unsigned integer values in an XMM register or 128-bit memory location and packs the 32-bit results in the destination XMM register.



Related Instructions

PHADDUBW, PHADDUBD, PHADDUBQ, PHADDUWQ, PHADDUDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

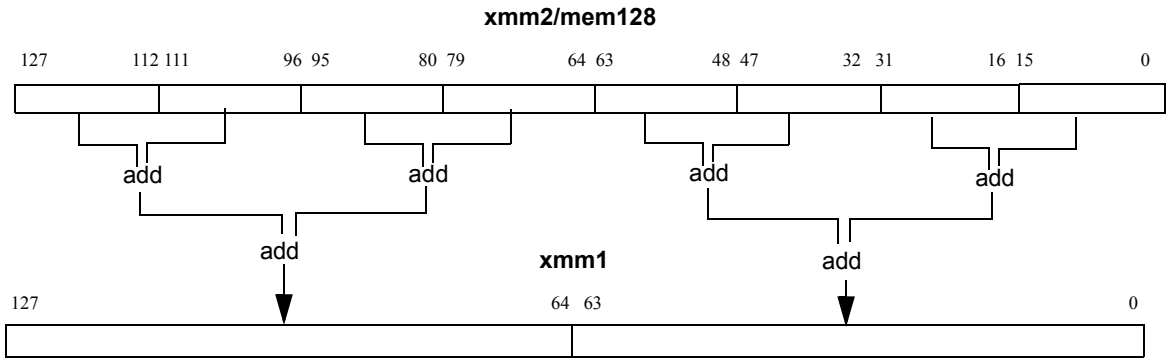
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDUWQ Packed Horizontal Add Unsigned Word to Quadword

Adds four successive 16-bit unsigned integer values from the second source operand and packs the results of the additions in a quadword 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 PHADDUWQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDUWQ *xmm1*, *xmm2*/*mem128*	0F 7A 57 /r	Adds four successive 16-bit unsigned integer values in an XMM register or 128-bit memory location and packs the 64-bit results in the destination XMM register.



Related Instructions

PHADDUBW, PHADDUBD, PHADDUBQ, PHADDUWD, PHADDUDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

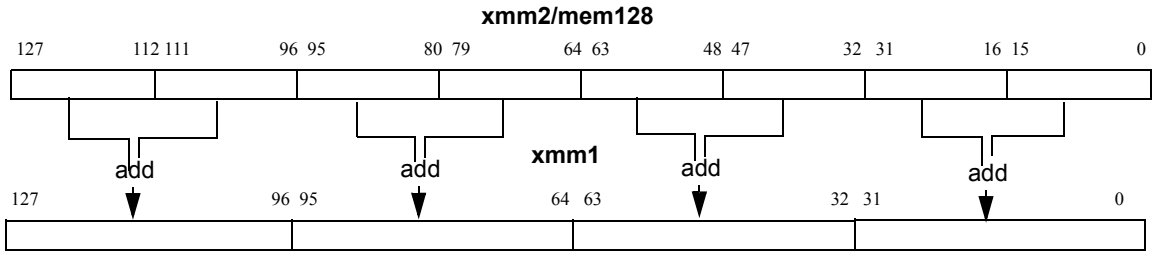
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDWD Packed Horizontal Add Signed Word to Signed Doubleword

Adds each adjacent pair of 16-bit signed integer values from the second source operand and packs the sign-extended results of the addition in a doubleword 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 PHADDWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDWD *xmm1*, *xmm2*/*mem128*	0F 7A 46 /r	Adds each adjacent pair of 16-bit signed integer values in an XMM register or 128-bit memory location and packs the 32-bit results in the destination XMM register.



Related Instructions

PHADDBW, PHADDBD, PHADDBQ, PHADDWQ, PHADDDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

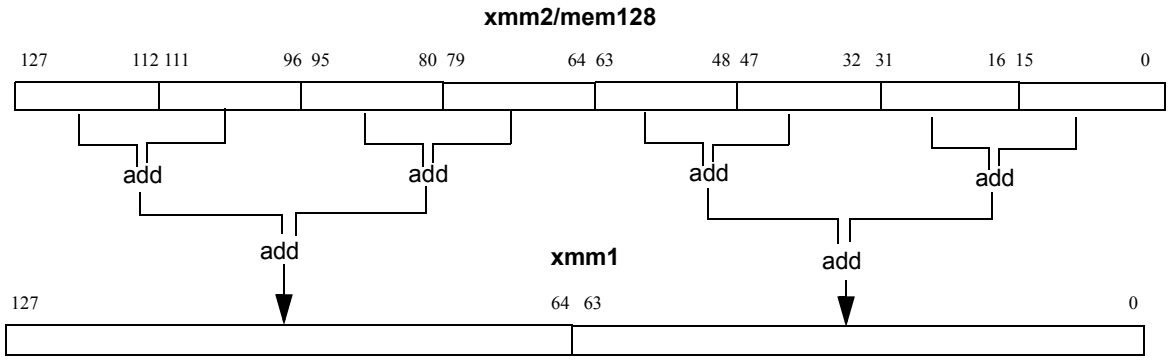
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHADDWQ Packed Horizontal Add Signed Word to Signed Quadword

Adds four successive 16-bit signed integer values from the second source operand and packs the sign-extended results of the additions in a quadword 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 PHADDWQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHADDWQ *xmm1*, *xmm2*/*mem128*	0F 7A 47 /r	Adds four successive 16-bit signed integer values in an XMM register or 128-bit memory location and packs the 64-bit results in the destination XMM register.



Related Instructions

PHADDBW, PHADDBD, PHADDBQ, PHADDWD, PHADDDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

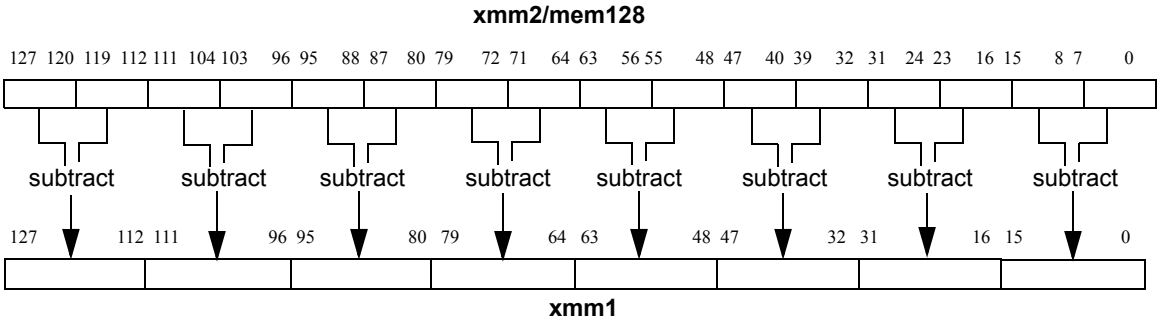
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHSUBBW Packed Horizontal Subtract Signed Byte to Signed Word

Subtracts the most significant signed integer byte from the least significant signed integer byte of each word from the second source operand and packs the sign-extended 16-bit integer result of each subtraction 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 PHSUBBW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHSUBBW *xmm1*, *xmm2*/*mem128*	0F 7A 61 /r	Subtracts the most significant byte from the least significant byte of each word in an XMM register or 128-bit memory location and packs the 16-bit results in the destination XMM register.



Related Instructions

PHSUBWD, PHSUBDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

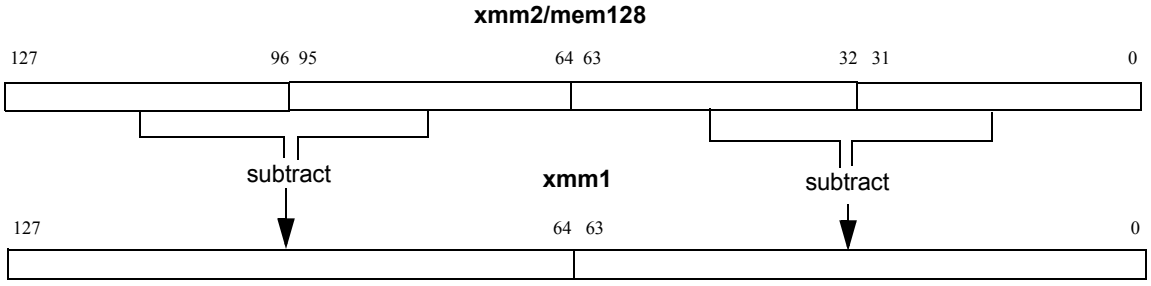
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHSUBDQ Packed Horizontal Subtract Signed Doubleword to Signed Quadword

Subtracts the most significant signed integer doubleword from the least significant signed integer doubleword of each quadword from the second source operand and packs the sign-extended 64-bit integer result of each subtraction 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 PHSUBDQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHSUBDQ *xmm1*, *xmm2*/*mem128*	0F 7A 63 /r	Subtracts the most significant doubleword from the least significant doubleword of each quadword in an XMM register or 128-bit memory location and packs the 64-bit results in the destination XMM register.



Related Instructions

PHSUBBW, PHSUBWD

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

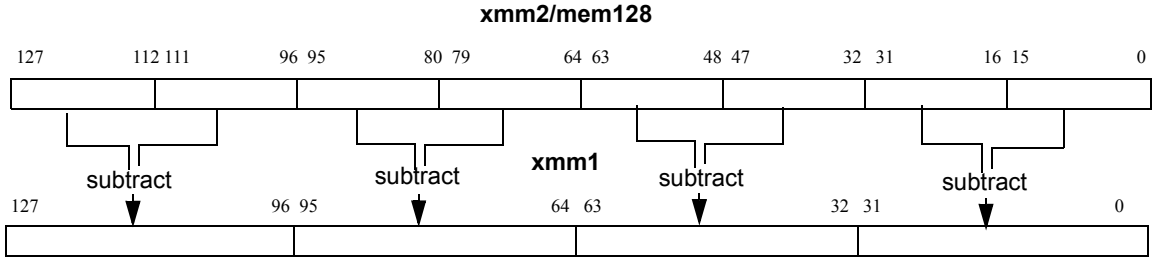
Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	X	X	X	The emulate bit (EM) of CR0 was set to 1.
	X	X	X	The operating-system FXSAVE/FXRSTOR support bit (OSFXSR) of CR4 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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PHSUBWD Packed Horizontal Subtract Signed Word to Signed Doubleword

Subtracts the most significant signed integer word from the least significant signed integer word of each doubleword from the second source operand and packs the sign-extended 32-bit integer result of each subtraction 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 PHSUBWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PHSUBWD *xmm1*, *xmm2*/*mem128*	0F 7A 62 /r	Subtracts the most significant word from the least significant word of each adjacent pair of 16-bit signed integer values in an XMM register or 128-bit memory location and packs the 32-bit results in the destination XMM register.



Related Instructions

PHSUBBW, PHSUBDQ

rFLAGS Affected

None

MXCSR FLAGS Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSDD Packed Multiply Accumulate Signed Doubleword to Signed Doubleword

Multiplies each packed 32-bit signed integer value in the first source operand by the corresponding packed 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the corresponding packed 32-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. The four results are written to the destination (accumulator) register.

No saturation is performed on the sum. If the result of the multiply causes non-zero values to be set in the upper 32 bits of the 64 bit product, they are ignored. If the result of the add overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set). In both cases, only the signed low-order 32 bits of the result are written in the destination.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSDD instruction requires four operands:

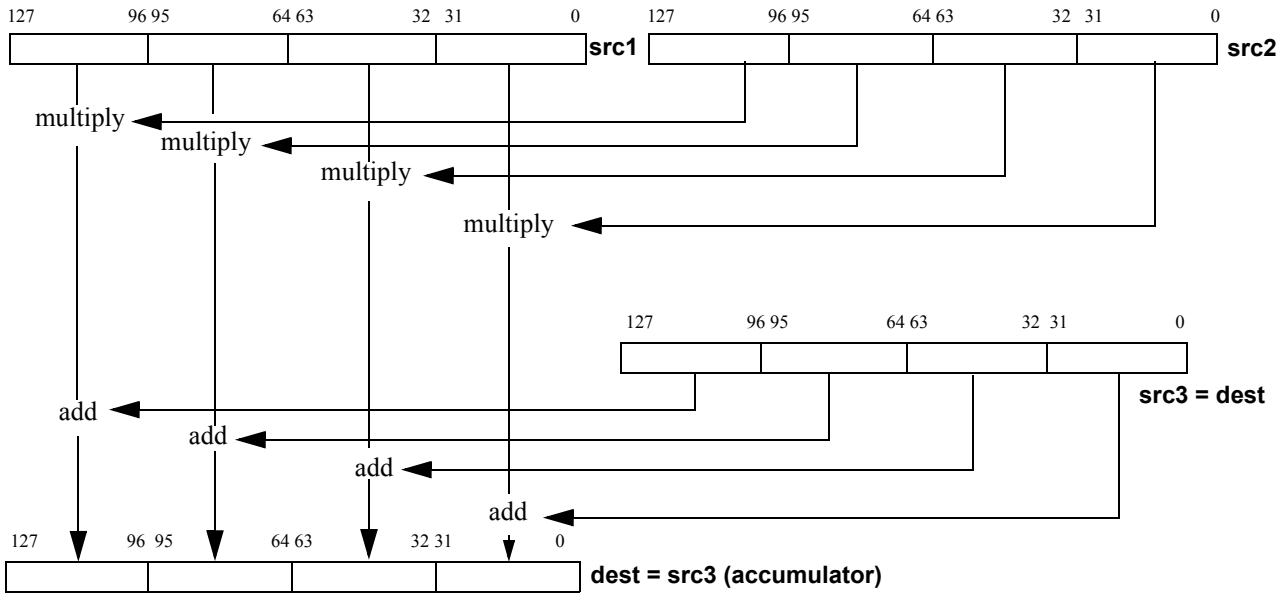
PMACSDD dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSDD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSDD *xmm1, xmm2, xmm3/mem128, xmm1*	0F 24 9E /r /drex0	Multiplies each packed 32-bit signed integer values in the second and third operands, then adds the 64-bit product to the corresponding packed 32-bit signed integer value in the fourth operand and writes the signed 32-bit result in the corresponding doubleword of the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+32)
{
     temp = xmm2[(31+i):i] * xmm3/mem128[(31+i):i];
     temp = xmm1[(31+i):i] + temp;
     xmm1[(31+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSDQH Packed Multiply Accumulate Signed High Doubleword to Signed Quadword

Multiplies the second 32-bit signed integer value of the first source operand by the second 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the low-order 64-bit signed integer value in the third source operand. Simultaneously, multiplies the fourth 32-bit signed integer value of the first source operand by the fourth 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the second 64-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. Both results are written to the destination register.

No saturation is performed on the sum. If the result of the add overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set). Only the low-order 64 bits of each result are written in the destination.

The destination register is an XMM register addressed by the DREX.dest field nd is identical to the third source register.

The PMACSDQH instruction requires four operands:

PMACSDQH dest, src1, src2, src3 *dest = src1* src2 + src3*

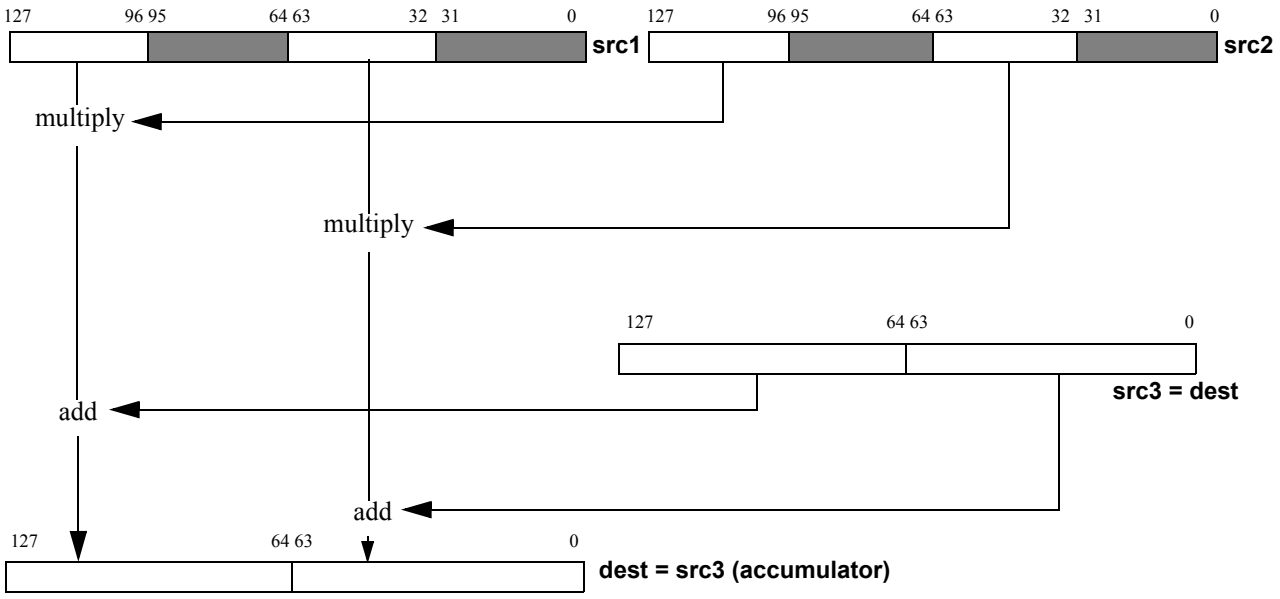
The PMACSDQH instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSDQH *xmm1, xmm2, xmm3/mem128, xmm1*	0F 24 9F /r /drex0	Multiplies the high doublewords in the second and third operand, then adds the signed 64-bit products to the signed 64-bit values in the fourth operand and writes the quadword results in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+64)
{
      temp = xmm2[(63+i):32+i] * xmm3/mem128[(63+i):32+i];
      temp = xmm1[(63+i):i] + temp;
      xmm1[(63+i):i] = temp;

}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSDQL Packed Multiply Accumulate Signed Low Doubleword to Signed Quadword

Multiplies the low-order 32-bit signed integer value of the first source operand by the low-order 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the low-order 64-bit signed integer value in the third source operand. Simultaneously, multiplies the third 32-bit signed integer value of the first source operand by the corresponding 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the second 64-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. Both results are written to the destination (accumulator) register.

No saturation is performed on the sum. If the result of the add overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set). Only the low-order 64 bits of each result are written in the destination.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSDQL instruction requires four operands:

PMACSDQL dest, src1, src2, src3 *dest = src1* src2 + src3*

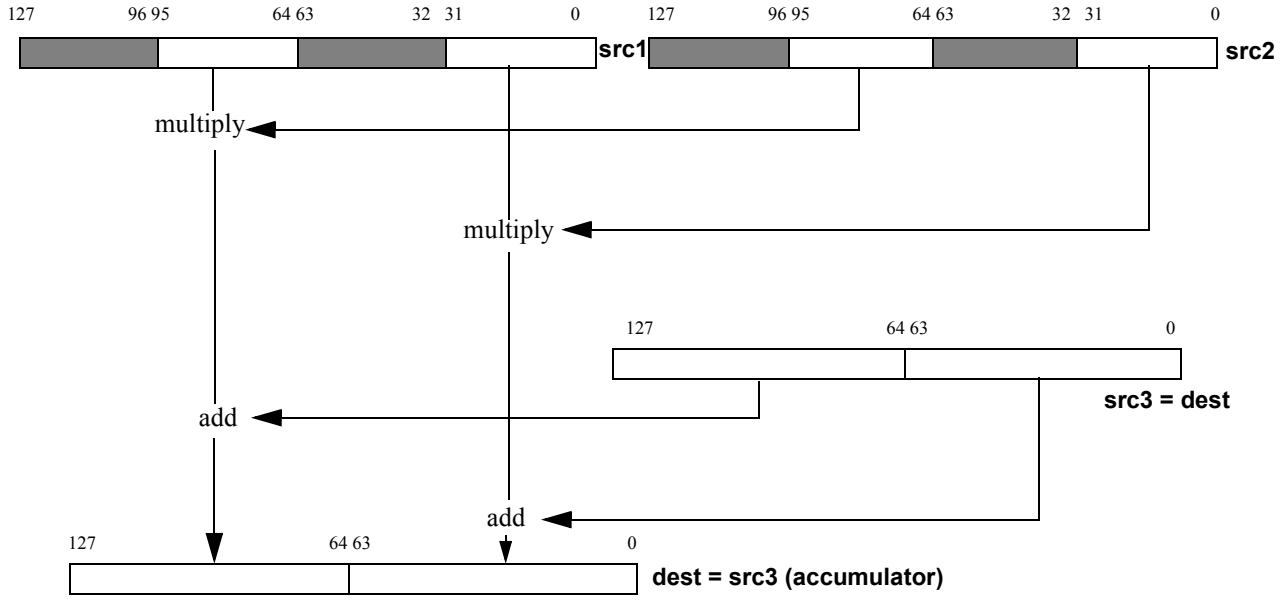
The PMACSDQL instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSDQL *xmm1*, *xmm2, xmm3/mem128, xmm1*	0F 24 97 /r /drex0	Multiplies the low doublewords in the second and third operands, then adds the signed 64-bit products to the signed 64-bit values in the fourth operand and writes the signed quadword results in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+64)
{
      temp = xmm2[(31+i):i] * xmm3/mem128[(31+i):i];
      temp = xmm1[(63+i):i] + temp;
      xmm1[(63+i):i] = temp;

}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSSDD Packed Multiply Accumulate Signed Doubleword to Signed Doubleword with Saturation

Multiplies each packed 32-bit signed integer value in the first source operand by the corresponding packed 32-bit signed integer value in the second source operand, then adds each 64-bit signed integer product to the corresponding packed 32-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. The four results are written to the destination (accumulator) register.

Out of range results of the addition are saturated to fit into a signed 32-bit integer. For each packed value in the destination, if the value is larger than the largest signed 32-bit integer, it is saturated to 7FFF_FFFFh, and if the value is smaller than the smallest signed 32-bit integer, it is saturated to 8000_0000h.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSSDD instruction requires four operands:

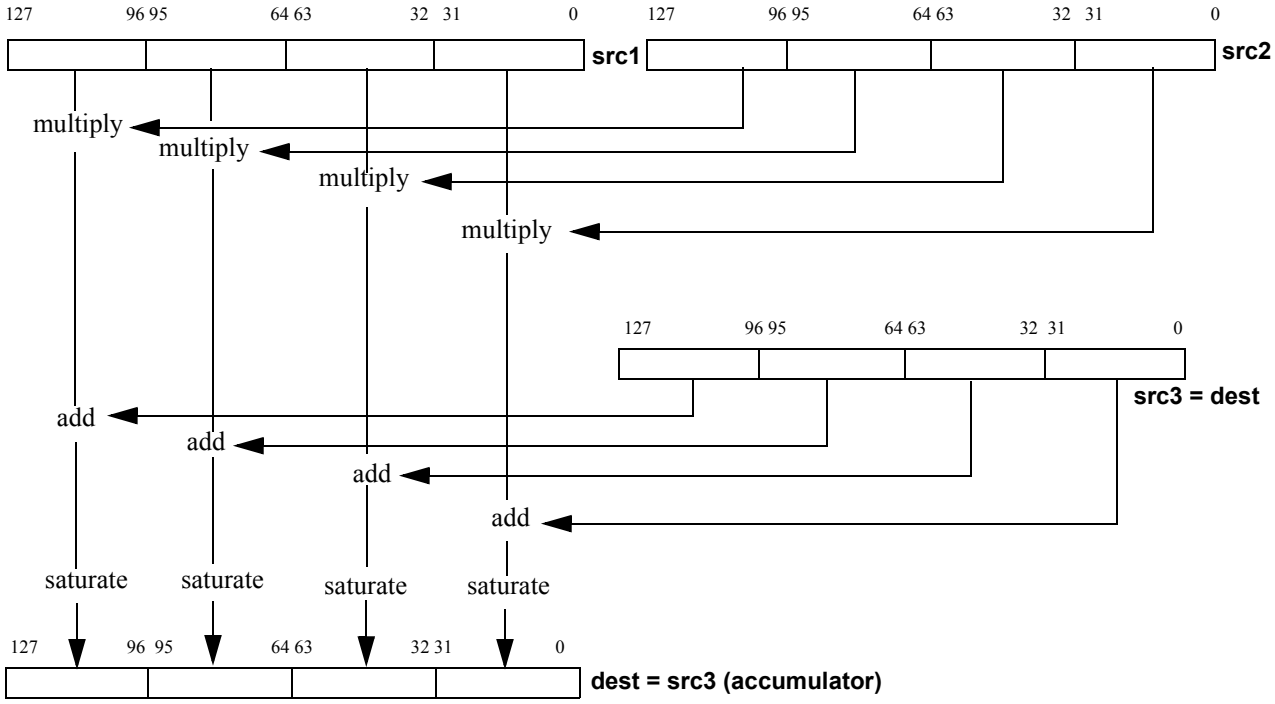
PMACSSDD dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSSDD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSSDD *xmm1, xmm2, xmm3/mem128, xmm1*	0F 24 8E /r /drex0	Multiplies each packed 32-bit signed integer values in the second and third operands, then adds each 64-bit product to the corresponding packed 32-bit signed integer value in the fourth operand and writes the signed saturated 32-bit result in the corresponding doubleword of the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+32)
{
     temp = xmm2[(31+i):i] * xmm3/mem128[(31+i):i];
     temp = xmm1[(31+i):i] + temp;
     if (temp < -2^31) temp = -2^31;
     if (temp > 2^31-1) temp = 2^31-1;
     xmm1[(31+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSSDQH Packed Multiply Accumulate Signed High Doubleword to Signed Quadword with Saturation

Multiplies the second 32-bit signed integer value of the first source operand by the second 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the low-order 64-bit signed integer value in the third source operand. Simultaneously, multiplies the fourth 32-bit signed integer value of the first source operand by the fourth 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the high-order 64-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. Both results are written to the destination (accumulator) register.

Out of range results of the addition are saturated to fit into a signed 64-bit integer. For each packed value in the destination, if the value is larger than the largest signed 64-bit integer, it is saturated to 7FFF_FFFF_FFFF_FFFFh, and if the value is smaller than the smallest signed 64-bit integer, it is saturated to 8000_0000_0000_0000h.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSSDQH instruction requires four operands:

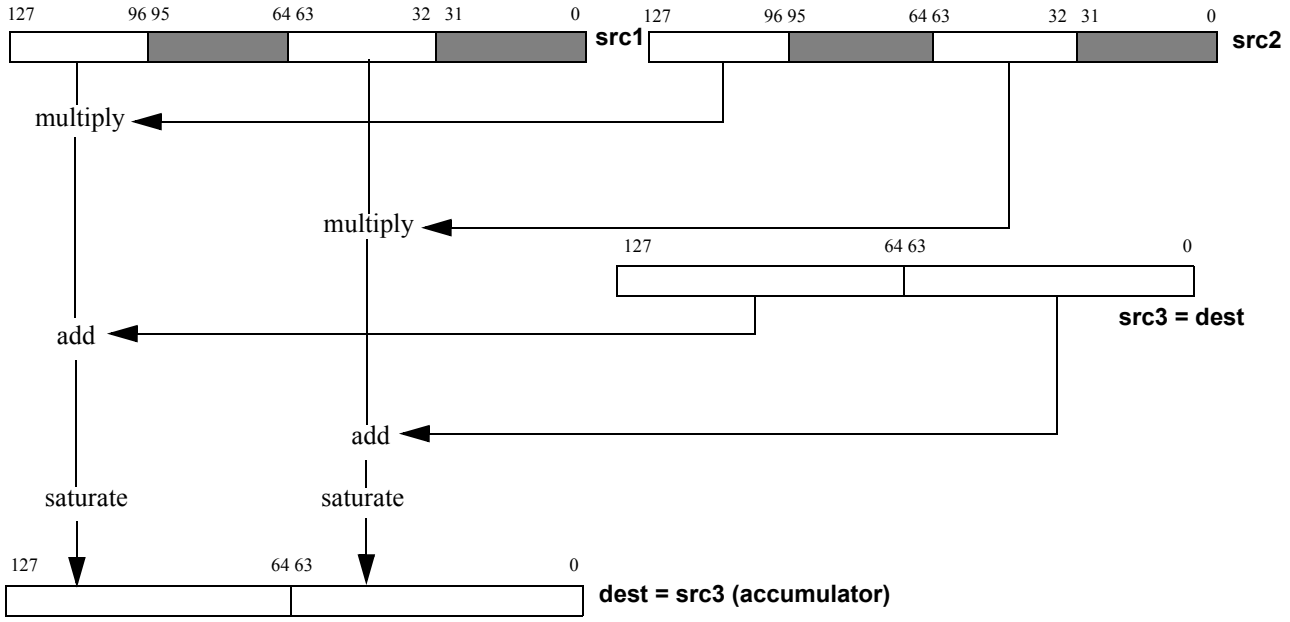
PMACSSDQH dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSSDQH instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSSDQH *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 8F /r /drex0	Multiplies the high doublewords in the second and third operands, then adds the signed products to the signed 64-bit integer values in the fourth operand. The quadword results are saturated and written to the destination register.

Action

```
for (i=0; i<128; i=i+64)
{
      temp = xmm2[(63+i):32+i] * xmm3/mem128[(63+i):32+i];
      temp = xmm1[(63+i):i] + temp;
      if (temp < -2^63) temp = -2^63;
      if (temp > (2^63 - 1)) temp = (2^63 - 1);
      xmm1[(63+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSSDQL Packed Multiply Accumulate Signed Low Doubleword to Signed Quadword with Saturation

Multiplies the low-order 32-bit signed integer value of the first source operand by the low-order 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the low-order 64-bit signed integer value in the third source operand. Simultaneously, multiplies the third 32-bit signed integer value of the first source operand by the third 32-bit signed integer value in the second source operand, then adds the 64-bit signed integer product to the high-order 64-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. Both results are written to the destination (accumulator) register.

Out of range results of the addition are saturated to fit into a signed 64-bit integer. For each packed value in the destination, if the value is larger than the largest signed 64-bit integer, it is saturated to 7FFF_FFFF_FFFF_FFFFh, and if the value is smaller than the smallest signed 64-bit integer, it is saturated to 8000_0000_0000_0000h.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSSDQL instruction requires four operands:

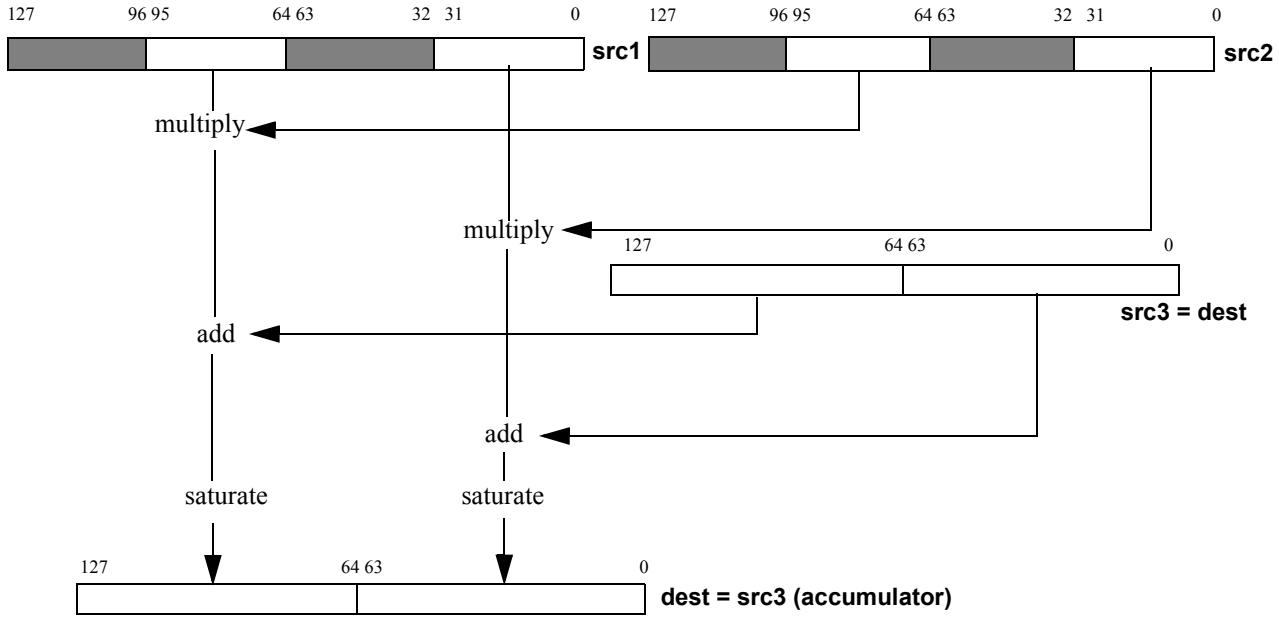
PMACSSDQL dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSSDQL instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSSDQL *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 87 /r /drex0	Multiplies the low doublewords in the second and third operands, then adds the 64-bit products to the signed 64-bit integer values in the fourth operand and writes the signed saturated quadword result in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+64)
{
      temp = xmm2[(31+i):i] * xmm3/mem128[(31+i):i];
      temp = xmm1[(63+i):i] + temp;
      if (temp < -2^63) temp = -2^63;
      if (temp > (2^63 - 1)) temp = (2^63 - 1);
      xmm1[(63+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSSWD Packed Multiply Accumulate Signed Word to Signed Doubleword with Saturation

Multiplies the odd-numbered packed 16-bit signed integer values in the first source operand by the corresponding packed 16-bit signed integer values in the second source operand, then adds the 32-bit signed integer products to the corresponding packed 32-bit signed integer values in the third source operand, which is the accumulator and is identical to the destination XMM register. The four results are written to the destination (accumulator) register.

Out of range results of the addition are saturated to fit into a signed 32-bit integer. For each packed value in the destination, if the value is larger than the largest signed 32-bit integer, it is saturated to 7FFF_FFFFh, and if the value is smaller than the smallest signed 32-bit integer, it is saturated to 8000_0000h.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSSWD instruction requires four operands:

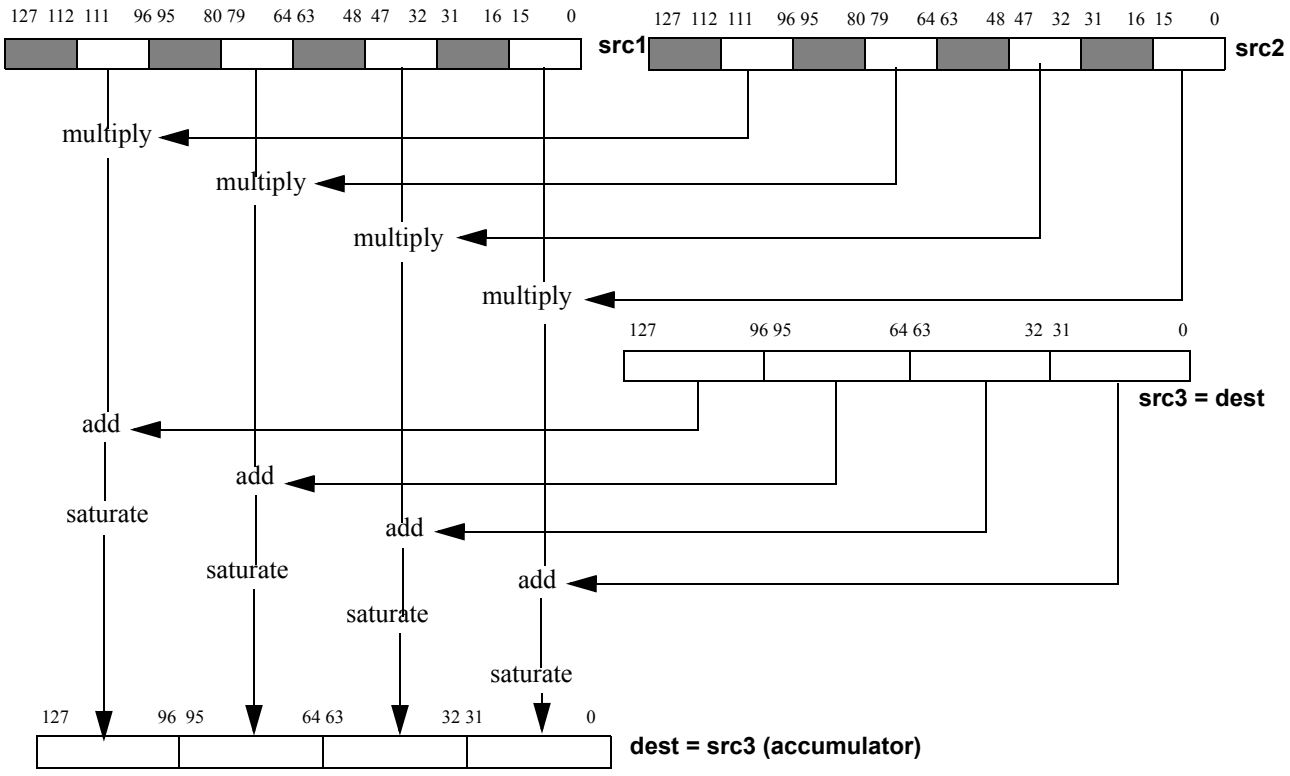
PMACSSWD dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSSWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSSWD *xmm1*, *xmm2, xmm3*/*mem128, xmm1*	0F 24 86 /r /drex0	Multiplies each odd-numbered packed 16-bit signed integer values in the second and third operands, then adds the 32-bit products to the corresponding packed 32-bit signed integer values in the fourth operand and writes the signed saturated 32-bit results in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+32)
{
     temp = xmm2[(15+i):i] * xmm3/mem128[(15+i):i];
     temp = xmm1[(31+i):i] + temp;
     if (temp < -2^31) temp = -2^31;
     if (temp > 2^31-1) temp = 2^31-1;
     xmm1[(31+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSSWW Packed Multiply Accumulate Signed Word to Signed Word with Saturation

Multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer value in the second source operand, then adds the 32-bit signed integer products to the corresponding packed 16-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. The eight results are written to the destination (accumulator) register.

Out of range results of the addition are saturated to fit into a signed 16-bit integer. 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 destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSSWW instruction requires four operands:

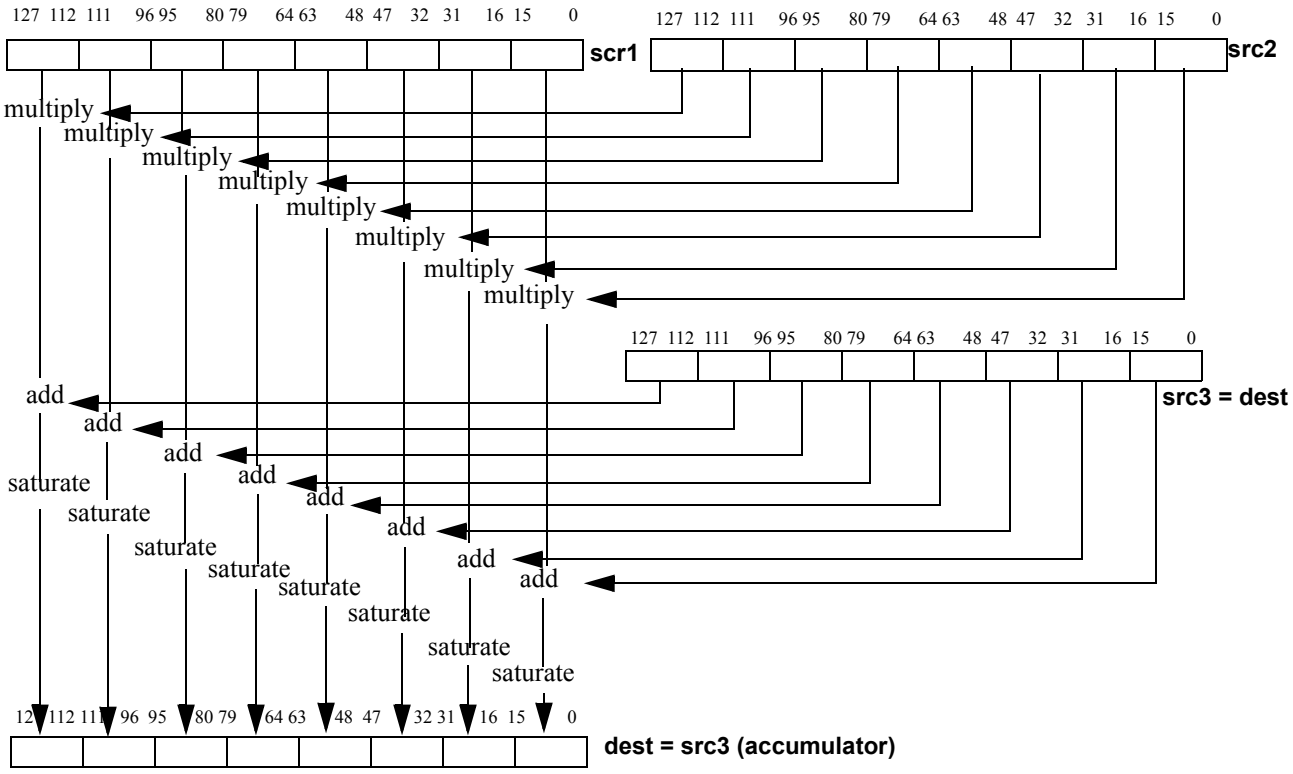
PMACSSWW dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSSWW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSSWW *xmm1*, *xmm2, xmm3/mem128, xmm1*	0F 24 85 /r /drex0	Multiplies packed 16-bit signed integer values in the second and third operands, then adds the 32-bit products to the corresponding packed 16-bit signed integer value in the fourth operand and writes the signed saturated 16-bit results in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+16)
{
     temp = xmm2[(15+i):i] * xmm3/mem128[(15+i):i];
     temp = xmm1[(15+i):i] + temp;
     if (temp < -32768) temp = -32768;
     if (temp > 32767) temp = 32767;
     xmm1[(15+i):i] = temp;
}
```



Related Instructions

PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSWD Packed Multiply Accumulate Signed Word to Signed Doubleword

Multiplies each odd-numbered packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer value in the second source operand, then adds the 32-bit signed integer products to the corresponding packed 32-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. The four results are written to the destination (accumulator) register.

If the result of the add overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set). Only the low-order 32 bits of the result are written in the destination.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSWD instruction requires four operands:

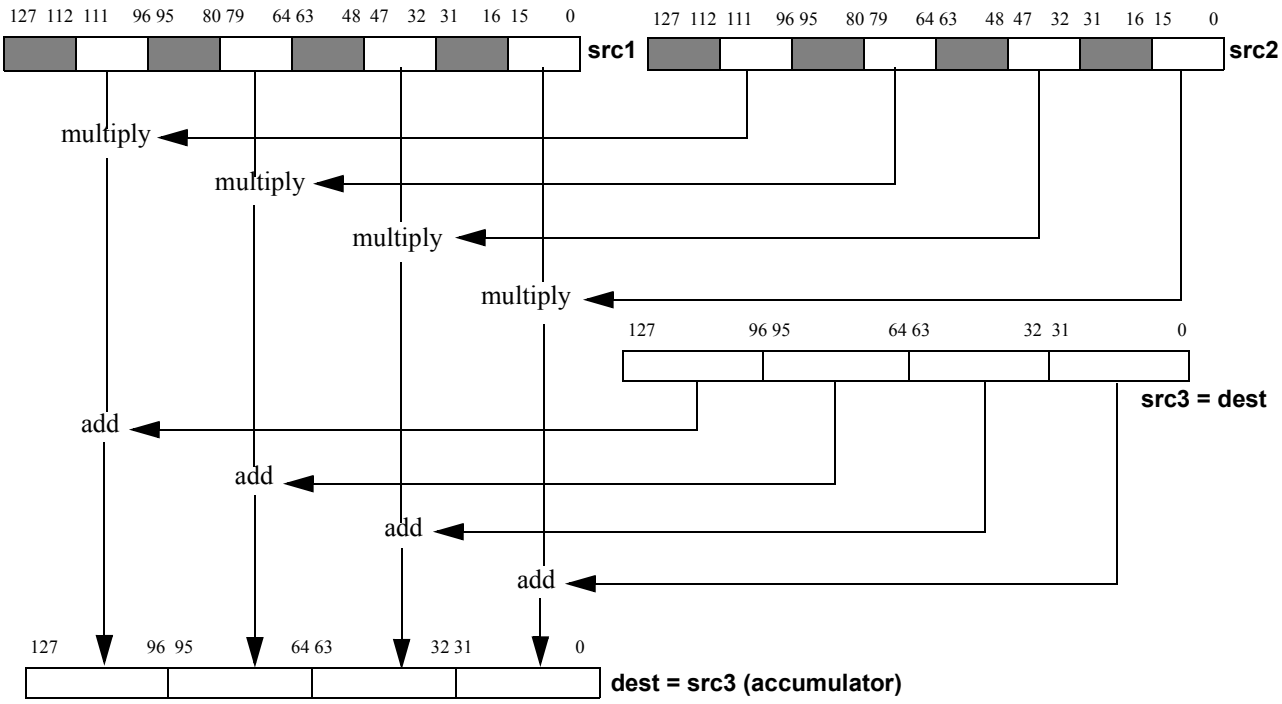
PMACSWD dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSWD *xmm1, xmm2, xmm3/mem128, xmm1*	0F 24 96 /r /drex0	Multiplies each odd-numbered packed 16-bit signed integer values in second and third operands, then adds each 32-bit product to the corresponding packed 32-bit signed integer value in the fourth operand and writes the signed 32-bit result in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+32)
{
     temp = xmm2[(15+i):i] * xmm3/mem128[(15+i):i];
     temp = xmm1[(31+i):i] + temp;
     xmm1[(31+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSSDD, PMACSDO, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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	The memory operand was not aligned on a 16-byte boundary while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMACSWW Packed Multiply Accumulate Signed Word to Signed Word

Multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer value in the second source operand, then adds each 32-bit signed integer product to the corresponding packed 16-bit signed integer value in the third source operand, which is the accumulator and is identical to the destination XMM register. The eight results are written to the destination (accumulator) register.

No saturation is performed on the sum. If the result of the multiply causes non-zero values to be set in the upper 16 bits of the 32 bit result, they are ignored. If the result of the add overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set). In both cases, only the signed low-order 16 bits of the result are written in the destination.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMACSWW instruction requires four operands:

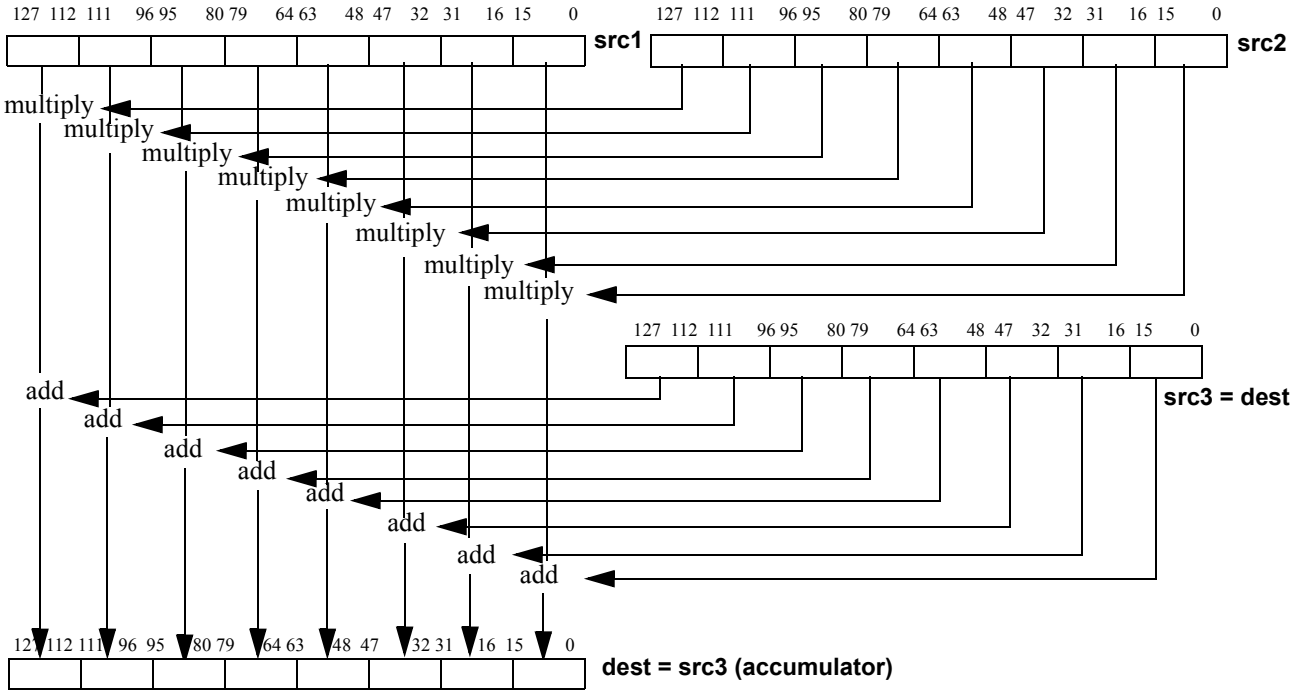
PMACSWW dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMACSWW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMACSWW *xmm1, xmm2, xmm3/mem128, xmm1*	0F 24 95 /r /drex0	Multiplies packed 16-bit signed integer values in the second and third operands, adds each 32-bit product to the corresponding packed 16-bit signed integer value in the fourth operand and writes the signed 16-bit results in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+16)
{
     temp = xmm2[(15+i):i] * xmm3/mem128[(15+i):i];
     temp = xmm1[(15+i):i] + temp;
     xmm1[(15+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMADCSSWD Packed Multiply, Add and Accumulate Signed Word to Signed Doubleword with Saturation

Multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer value in the second source operand, then adds the 32-bit signed integer products of the even-odd adjacent words. Each resulting sum is then added to the corresponding packed 32-bit signed integer value in the third source operand, which is the accumulator, as is identical to the destination XMM register. The four results are written to the destination (accumulator) register.

Out of range results of the addition are saturated to fit into a signed 32-bit integer. For each packed value in the destination, if the value is larger than the largest signed 32-bit integer, it is saturated to 7FFF_FFFFh, and if the value is smaller than the smallest signed 32-bit integer, it is saturated to 8000_0000h.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMADCSSWD instruction requires four operands:

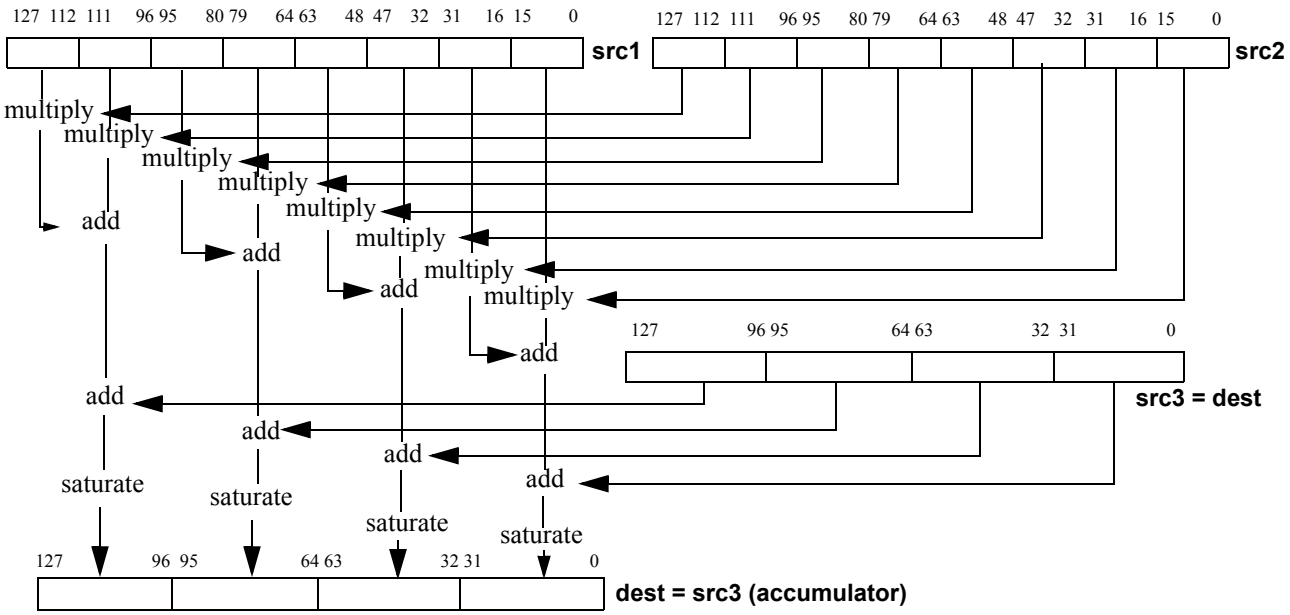
PMADCSSWD dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMADCSSWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMADCSSWD *xmm1*, *xmm2*, *xmm3*/*mem128, xmm1*	0F 24 A6 /r /drex0	Multiplies packed signed 16-bit integer values in the second and third operands, then adds the 32-bit products of the even-odd adjacent words together. Finally, adds their sum to the corresponding packed 32-bit signed integer value in the fourth operand and writes the signed saturated 32-bit results in the destination (xmm1 register).

Action

```
for (i=0; i<128; i=i+32)
{
     temp1 = xmm2[(15+i):i] * xmm3/mem128[(15+i):i];
     temp2 = xmm2[(31+i):16+i] * xmm3/mem128[(31+i):16+i];
     temp = temp1 + temp2
     temp = xmm1[(31+i):i] + temp;
     if (temp < -2^31) temp = -2^31;
     if (temp > 2^31-1) temp = 2^31-1;
     xmm1[(31+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PMADCSWD Packed Multiply Add and Accumulate Signed Word to Signed Doubleword

Multiplies each packed 16-bit signed integer value in the first source operand by the corresponding packed 16-bit signed integer value in the second source operand, then adds the 32-bit signed integer products of the even-odd adjacent words together and adds their sum to the corresponding packed 32-bit signed integer values in the third source operand, which is the accumulator and is identical to the destination XMM register. The four results are written to the destination (accumulator) register.

No saturation is performed on the sum. If the result of the adds overflows, the carry is ignored (neither the overflow nor carry bit in rFLAGS is set). Only the signed 32-bits of the result are written in the destination.

The destination register is an XMM register addressed by the DREX.dest field and is identical to the third source register.

The PMADCSWD instruction requires four operands:

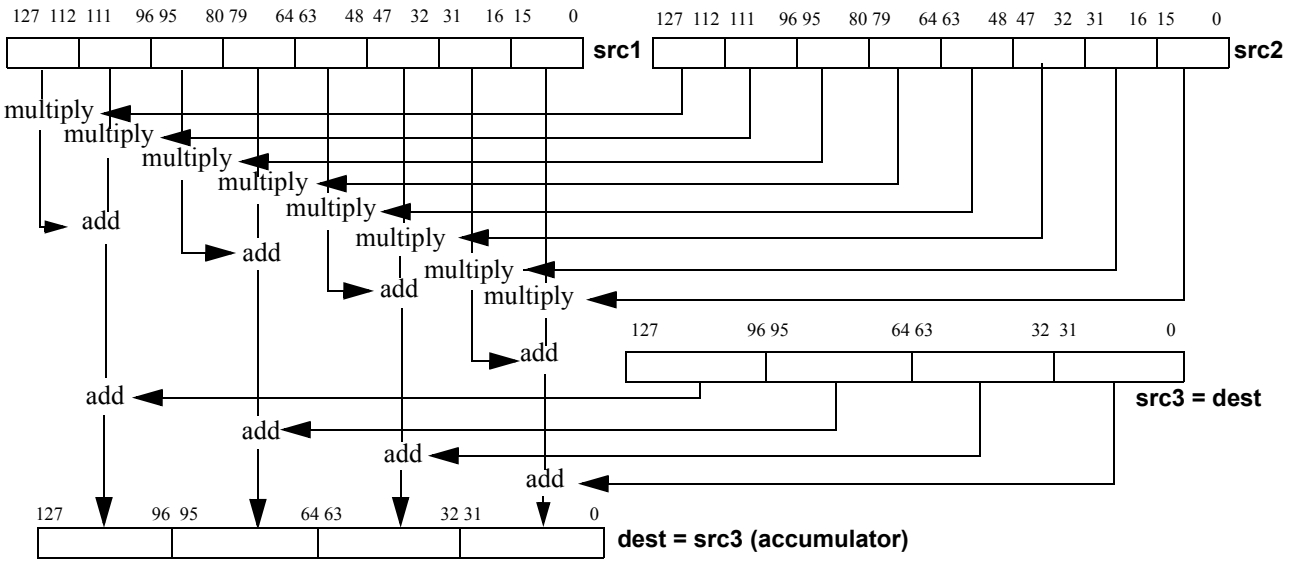
PMADCSWD dest, src1, src2, src3 *dest = src1* src2 + src3*

The PMADCSWD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PMADCSWD *xmm1*, *xmm2*, *xmm3*/*mem128, xmm1*	0F 24 B6 /r /drex0	Multiplies packed signed 16-bit integer values in the second and third operands, then adds the 32-bit products of the even-odd adjacent words together. Finally, adds their sum to the corresponding packed 32-bit signed integer value in the fourth operand and writes the signed 32-bit results in the destination (xmm1register).

Action

```
for (i=0; i<128; i=i+32)
{
     temp1 = xmm2[(15+i):i] * xmm3/mem128[(15+i):i];
     temp2 = xmm2[(31+i):16+i] * xmm3/mem128[(31+i):16+i];
     temp = temp1 + temp2
     temp = xmm1[(31+i):i] + temp;
     xmm1[(31+i):i] = temp;
}
```



Related Instructions

PMACSSWW, PMACSWW, PMACSSWD, PMACSWD, PMACSSDD, PMACSDD, PMACSSDQL, PMACSSDQH, PMACSDQL, PMACSDQH, PMADCSSWD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PPERM Packed Permute Bytes

Moves any of the 32-packed bytes in the source operands to each byte of the destination XMM register. Each byte of the result can optionally have a logical operation applied to it.

The 32-byte source operand consists of the second source operand (*src2*) concatenated with the first source operand (*src1*). The third source operand (*src3*) contains control bytes specifying the source byte and the logical operation for each destination byte.

The destination register is an XMM register addressed by the DREX.dest field.

The PPERM instruction requires four operands:

PPERM dest, src1, src2,src3

For each byte of the 16-byte result, the corresponding byte in *src3* is used as follows:

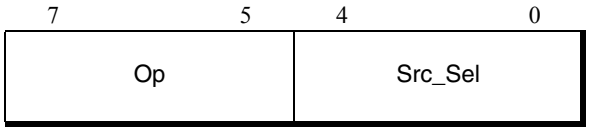
- bits 4:0 of *src3* select one of the 32 bytes from src2:src1
- bits 7:5 of *src3* select the logical operation applied.

The PPERM instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

The control byte is defined in Table 2-3, “PPERM Control Byte”, on page 194.

Mnemonic	Opcode	Description
PPERM *xmm1*, *xmm1, xmm2, xmm3*/*mem128*	0F 24 23 /r /drex0	For each byte position of the 16-byte result, uses corresponding control byte in fourth operand to perform logical operation on one of 32 bytes from the second and third source operands and writes result in destination (xmm1 register).
PPERM *xmm1*, *xmm1, xmm3*/*mem128, xmm2*	0F 24 23 /r /drex1	
PPERM *xmm1*, *xmm2*, *xmm3*/*mem128, xmm1*	0F 24 27 /r /drex0	
PPERM *xmm1*, *xmm3*/*mem128, xmm2, xmm1*	0F 24 27 /r /drex1	

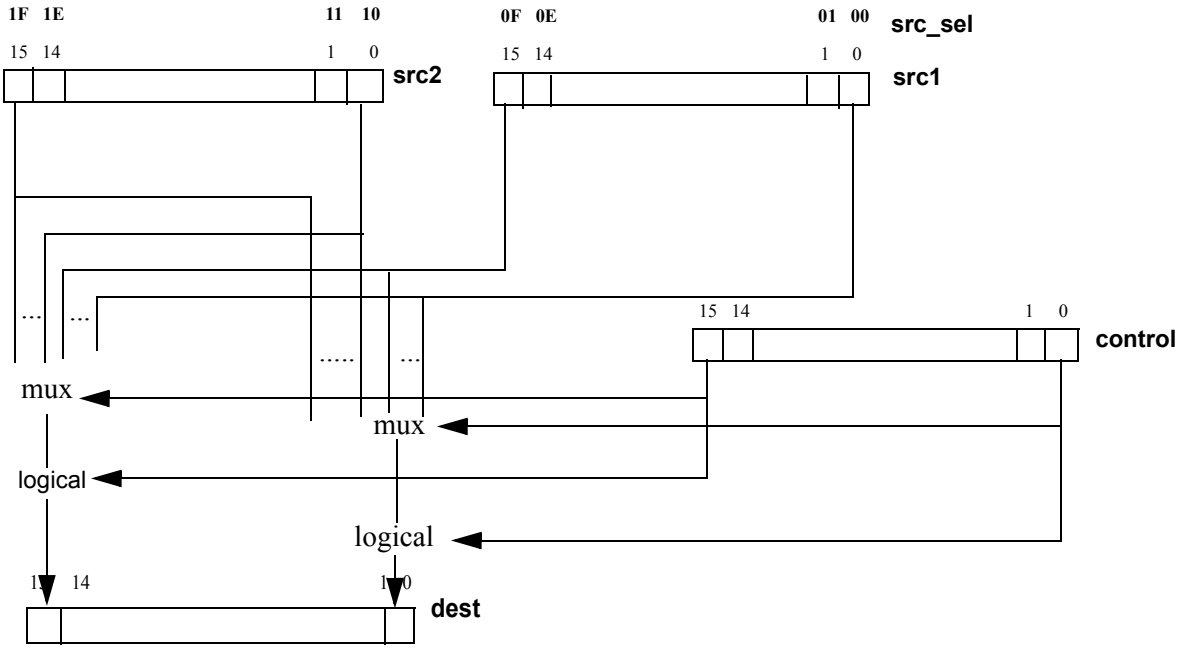
Table 2-3. PPERM Control Byte



Control Byte				
Bits	Description			
7:5	**Op -** Defines the logical operation performed on the selected operand.			
	OP	**Operation**		
	000	Source byte (no logical operation)		
	001	Invert source byte		
	010	Bit reverse of source byte		
	011	Bit reverse of inverted source byte		
	100	0x00		
	101	0xFF		
	110	Most significant bit of source byte replicated in all bit positions.		
	111	Invert most significant bit of source byte and replicate in all bit positions.		
4:0	**Src_Sel** Selects the source byte to be operated on.			
	Src_Sel	**Source Selected**	**Src_Sel**	**Source Selected**
	00000	Src1[7:0]	10000	Src2[7:0]
	00001	Src1[15:8]	10001	Src2[15:8]
	00010	Src1[23:16]	10010	Src2[23:16]
	00011	Src1[31:24]	10011	Src2[31:24]
	00100	Src1[39:32]	10100	Src2[39:32]
	00101	Src1[47:40]	10101	Src2[47:40]
	00110	Src1[55:48]	10110	Src2[55:48]
	00111	Src1[63:56]	10111	Src2[63:56]
	01000	Src1[71:64]	11000	Src2[71:64]
	01001	Src1[79:72]	11001	Src2[79:72]
	01010	Src1[87:80]	11010	Src2[87:80]
	01011	Src1[95:88]	11011	Src2[95:88]
	01100	Src1[103:96]	11100	Src2[103:96]
	01101	Src1[111:104]	11101	Src2[111:104]
	01110	Src1[119:112]	11110	Src2[119:112]
	01111	Src1[127:120]	11111	Src2[127:120]

Action

```
for (i=0; i<16; i=++)
  dest[i]:= control[i].op (src1|src2)control[i].src_sel;
```



Related Instructions

PSHUFHW, PSHUFD, PSHUFLW, PSHUFW, PERMPS, PERMPD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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	The memory operand was not aligned on a 16-byte boundary while MXCSR.MM=0.
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 while MXCSR.MM=1.

PROTB Packed Rotate Bytes

Rotates each byte of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding byte of the destination.

If the count value is positive, bits are rotated to the left (toward the more significant bit positions). The bits rotated out left of the most significant bit are rotated back in at the right end (least-significant bit) of the byte.

If the count value is negative, bits are rotated to the right (toward the least significant bit positions). The bits rotated out right of the least significant bit are rotated back in at the left end (most-significant bit) of the byte.

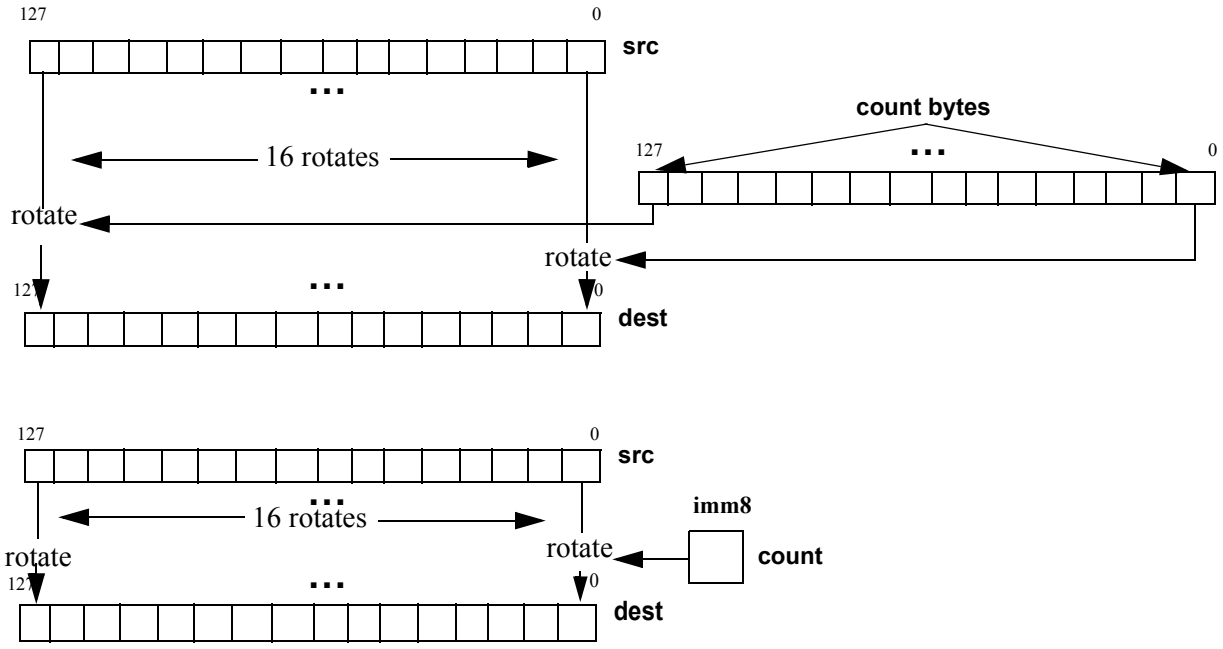
The rotate amount is stored in two's-complement form. The count is modulo 8.

The PROTB instruction has two variants:

- *PROTB dest, src, variable-count*—The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field. Each byte of the source operand is rotated by the amount specified in the corresponding byte of the *variable-count* operand, which is an XMM register or 128-bit memory operand.
- *PROTB dest, src, fixed-count*—Each byte of the source operand is rotated by the identical amount, as specified by the immediate byte *fixed-count* operand.

The PROTB instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PROTB *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 40 /r /drex0	Rotates each byte of the source operand (2nd operand) by the amount specified in the signed value of the corresponding count byte (3rd operand) and writes the result in the corresponding byte of the destination.
PROTB *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 40 /r /drex1	
PROTB *xmm1*, *xmm2*/*mem128, imm8*	0F 7B 40 /r ib	Rotates each byte of the source operand (2nd operand) by the (same) amount specified in the signed value of the count byte (immediate operand which is the 3rd operand) and writes the result in the corresponding byte of the destination.



Related Instructions

PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PROTD Packed Rotate Doublewords

Rotates each of the four doublewords of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding doubleword of the destination.

If the count value is positive, bits are rotated to the left (toward the more significant bit positions). The bits rotated out left of the most significant bit are rotated back in at the right end (least-significant bit) of the doubleword.

If the count value is negative, bits are rotated to the right (toward the least significant bit positions). The bits rotated out right of the least significant bit are rotated back in at the left end (most-significant bit) of the doubleword.

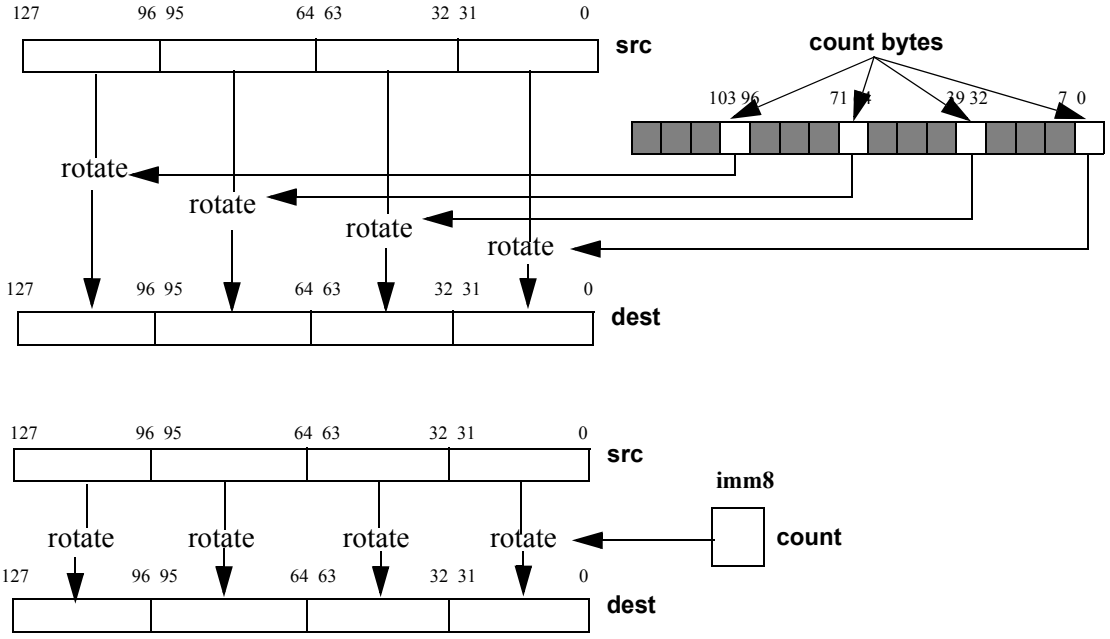
The rotate amount is stored in two's-complement form. The count is modulo 32.

The PROTD instruction has two variants:

- *PROTD dest, src, variable-count*—The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field. Each doubleword of the source operand is rotated by the amount specified in the corresponding doubleword of the *variable-count* operand, which is an XMM register or 128-bit memory operand.
- *PROTD dest, src, fixed-count*—Each doubleword of the source operand is rotated by the identical amount, as specified by the immediate byte *fixed-count* operand.

The PROTW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PROTD *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 42 /r /drex0	Rotates each doubleword of the source operand (2nd operand) by the amount specified in the low-order byte of the corresponding count doubleword (3rd operand) and writes the result in the corresponding doubleword of the destination.
PROTD *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 42 /r /drex1	
PROTD *xmm1, xmm2*/*mem128*, *imm8*	0F 7B 42 /r ib	Rotates each doubleword of the source operand (2nd operand) by the (same) amount specified in the signed value of the count byte (immediate operand which is the 3rd operand) and writes the result in the corresponding doubleword of the destination.



Related Instructions

PROTB, PROTW, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PROTQ Packed Rotate Quadwords

Rotates each of the quadwords of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding quadword of the destination.

If the count value is positive, bits are rotated to the left (toward the more significant bit positions). The bits rotated out left of the most significant bit are rotated back in at the right end (least-significant bit) of the quadword.

If the count value is negative, bits are rotated to the right (toward the least significant bit positions). The bits rotated out right of the least significant bit are rotated back in at the left end (most-significant bit) of the quadword.

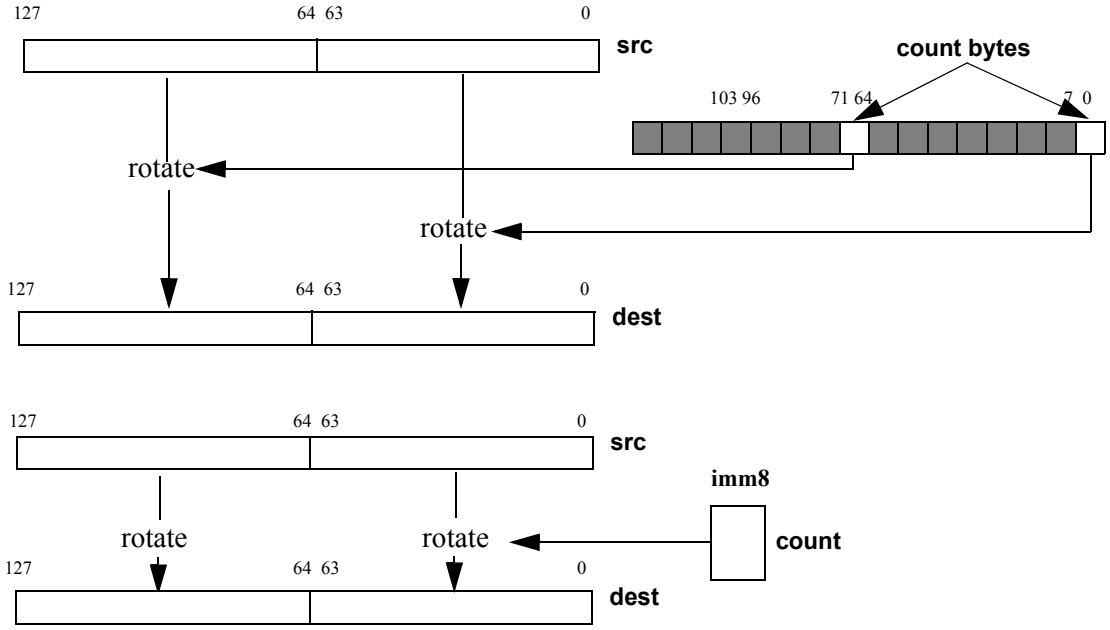
The rotate amount is stored in two's-complement form. The count is modulo 64.

The PROTQ instruction has two variants:

- *PROTQ dest, src, variable-count*—The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field. Each quadword of the source operand is rotated by the amount specified in the corresponding quadword of the *variable-count* operand, which is an XMM register or 128-bit memory operand.
- *PROTQ dest, src, fixed-count*—Each quadword of the source operand is rotated by the identical amount, as specified by the immediate byte *fixed-count* operand.

The PROTQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PROTQ *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 43 /r /drex0	Rotates each quadword of the source operand (2nd operand) by the amount specified in the low-order byte of the corresponding quadword in the third operand and writes the result in the corresponding quadword of the destination.
PROTQ *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 43 /r /drex1	
PROTQ *xmm1, xmm2*/*mem128*, *imm8*	0F 7B 43 /r ib	Rotates each quadword of the source operand (2nd operand) by the (same) amount specified in the signed value of the count byte (immediate operand which is the 3rd operand) and writes the result in the corresponding quadword of the destination.



Related Instructions

PROTB, PROTW, PROTD, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PROTW Packed Rotate Words

Rotates each of the eight words of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding word of the destination.

If the count value is positive, bits are rotated to the left (toward the more significant bit positions). The bits rotated out left of the most significant bit are rotated back in at the right end (least-significant bit) of the word.

If the count value is negative, bits are rotated to the right (toward the least significant bit positions). The bits rotated out right of the least significant bit are rotated back in at the left end (most-significant bit) of the word.

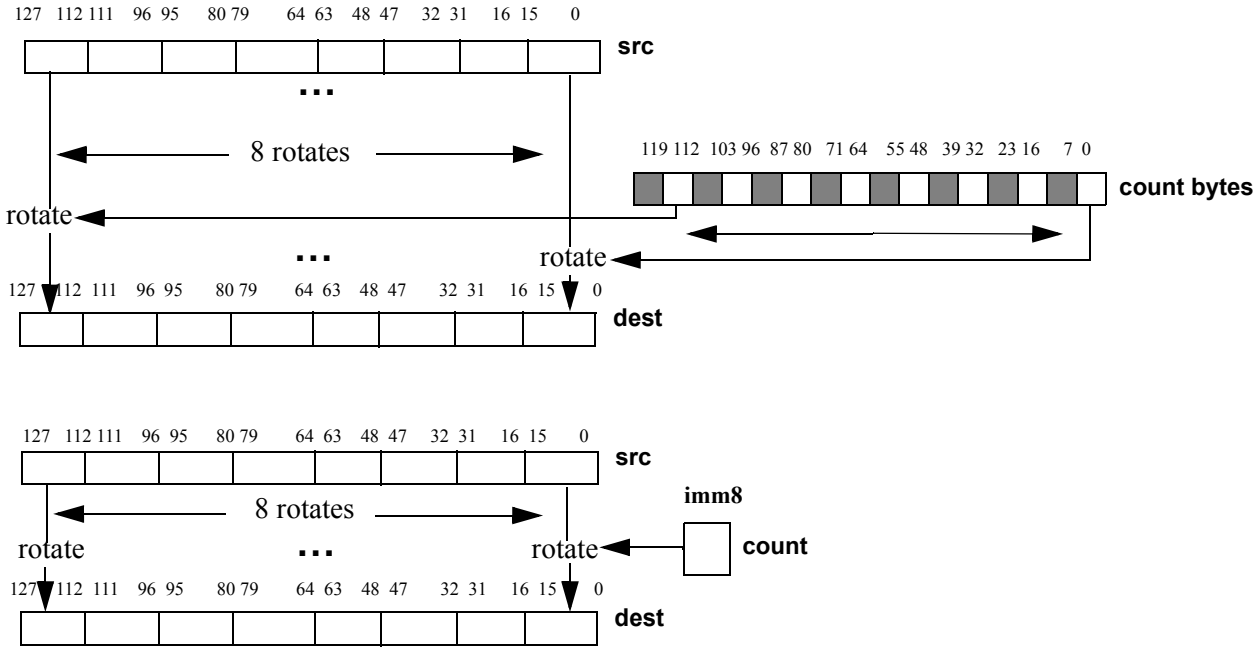
The rotate amount is stored in two's-complement form. The count is modulo 16.

The PROTW instruction has two variants:

- *PROTW dest, src, variable-count*—The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field. Each word of the source operand is rotated by the amount specified in the corresponding word of the *variable-count* operand, which is an XMM register or 128-bit memory operand.
- *PROTW dest, src, fixed-count*—Each word of the source operand is rotated by the identical amount, as specified by the immediate byte *fixed-count* operand.

The PROTW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PROTW *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 41 /r /drex0	Rotates each word of the source operand (2nd operand) by the amount specified in the low-order byte of the corresponding word in the third operand and writes the result in the corresponding word of the destination.
PROTW *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 41 /r /drex1	
PROTW *xmm1, xmm2*/*mem128*, *imm8*	0F 7B 41 /r ib	Rotates each word of the source operand (2nd operand) by the (same) amount specified in an immediate byte and writes the result in the corresponding word of the destination.



Related Instructions

PROTB, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHAB Packed Shift Arithmetic Bytes

Shifts each byte of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding byte of the destination.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the byte.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). The most significant bit is replicated and shifted in at the left end (most-significant bit) of the byte.

The shift amount is stored in two's-complement form. The count is modulo 8.

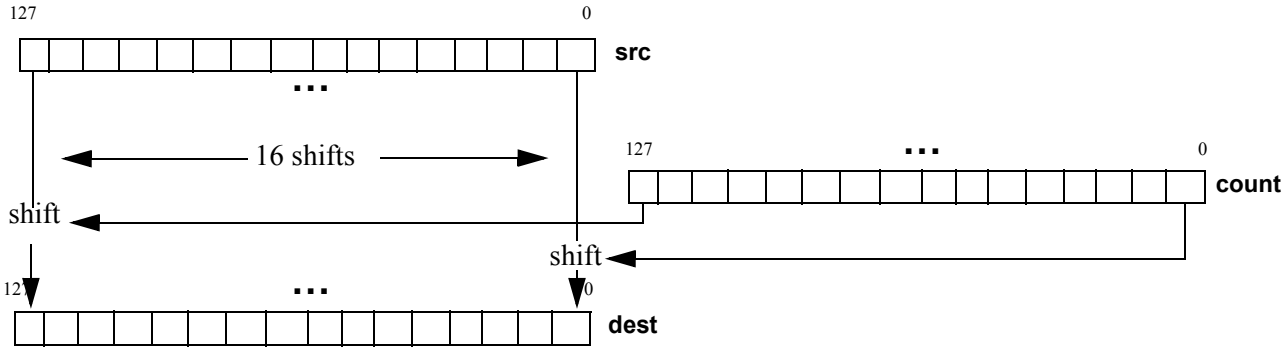
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHAB instruction requires three operands:

PSHAB dest, src, count

The PSHAB instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHAB *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 48 /r /drex0	Shifts each byte of second operand by an amount specified in the corresponding byte in the third operand and writes the result in the corresponding byte of the destination (xmm1 register).
PSHAB *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 48 /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHAD Packed Shift Arithmetic Doublewords

Shifts each of the four doublewords of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding doubleword of the destination.

The count byte is located in the low-order byte of the corresponding doubleword of the count operand.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the doubleword.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). The most significant bit is replicated and shifted in at the left end (most-significant bit) of the doubleword.

The shift amount is stored in two's-complement form. The count is modulo 32.

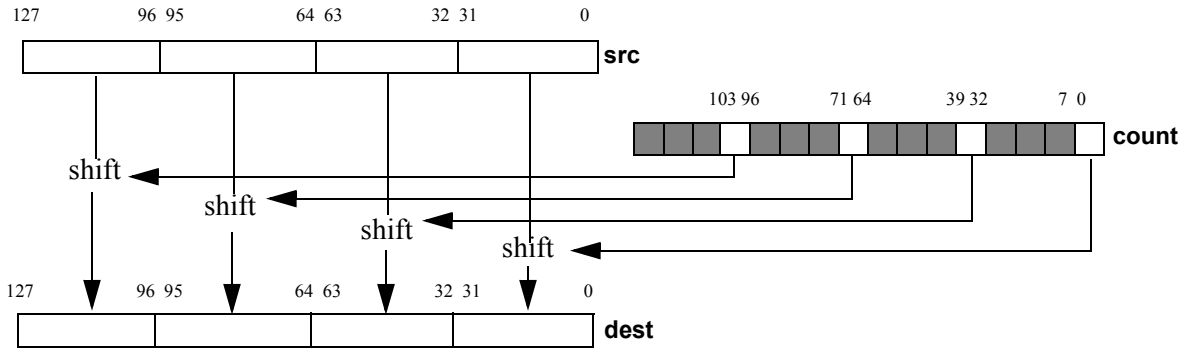
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHAD instruction requires three operands:

PSHAD dest, src, count

The PSHAD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHAD *xmm1*, *xmm2, xmm3/mem128*	0F 24 4A /r /drex0	Shifts each doubleword of second operand by an amount specified in the low-order byte of the corresponding doubleword of third operand and writes the result in the corresponding doubleword of the destination (xmm1 register).
PSHAD *xmm1*, *xmm3/mem128, xmm2*	0F 24 4A /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHAQ Packed Shift Arithmetic Quadwords

Shifts the two quadwords of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding quadword of the destination.

The count byte is located in the low-order byte of the corresponding quadword of the count operand.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the quadword.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). The most significant bit is replicated and shifted in at the left end (most-significant bit) of the quadword.

The shift amount is stored in two's-complement form. The count is modulo 64.

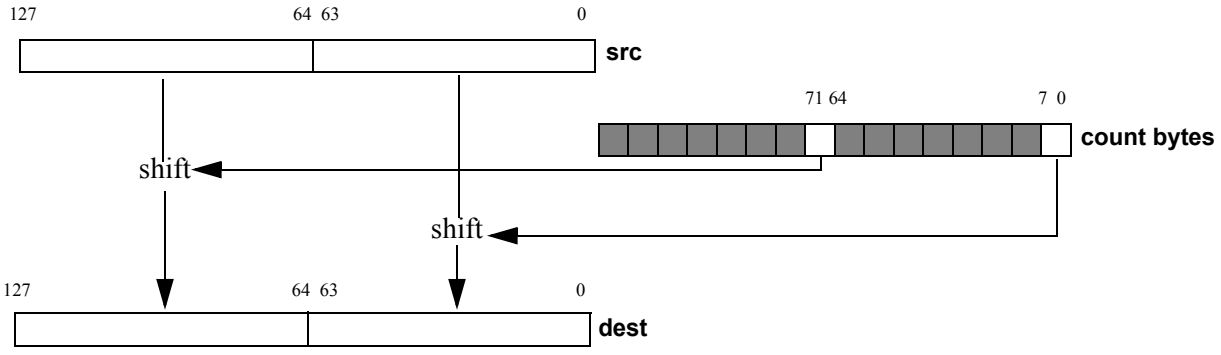
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHAQ instruction requires:

PSHAQ dest, src, count

The PSHAQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHAQ *xmm1, xmm2, xmm3/mem128*	0F 24 4B /r /drex0	Shifts each quadword of second operand by an amount specified in the low-order byte of the corresponding quadword in the third operand and writes the result in the corresponding quadword of the destination (xmm1 register).
PSHAQ *xmm1, xmm3/mem128, xmm2*	0F 24 4B /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHAW Packed Shift Arithmetic Words

Shifts each of the eight words of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding word of the destination.

The count byte is located in the low-order byte of the corresponding word of the count operand.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the word.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). The most significant bit is replicated and shifted in at the left end (most-significant bit) of the word.

The shift amount is stored in two's-complement form. The count is modulo 16.

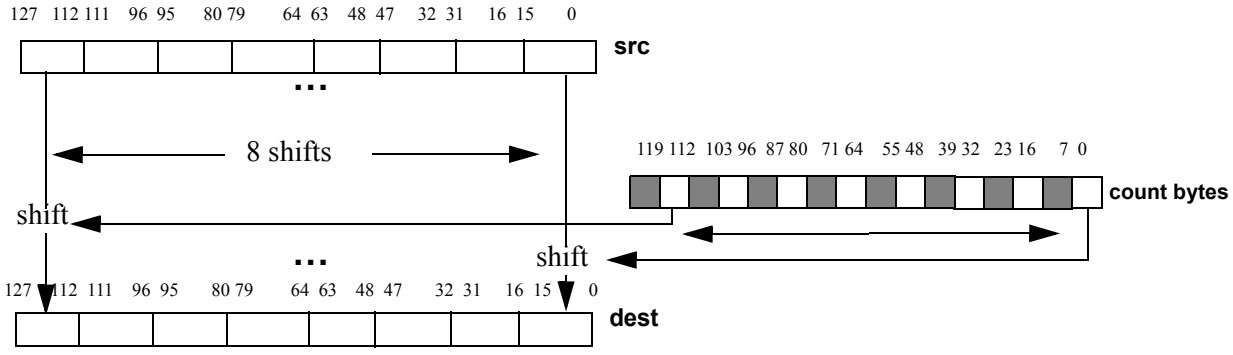
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHAW instruction requires three operands:

PSHAW dest, src, count

The PSHAW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHAW *xmm1*, *xmm2, xmm3/mem128*	0F 24 49 /r /drex0	Shifts each word of second operand by an amount specified in the low-order byte of the corresponding word in the third operand and writes the result in the corresponding word of the destination (xmm1 register).
PSHAW *xmm1*, *xmm3/mem128, xmm2*	0F 24 49 /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHLB Packed Shift Logical Bytes

Shifts each byte of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding byte of the destination.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the byte.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). Zeros are shifted in at the left end (most-significant bit) of the byte.

The shift amount is stored in two's-complement form. The count is modulo 8.

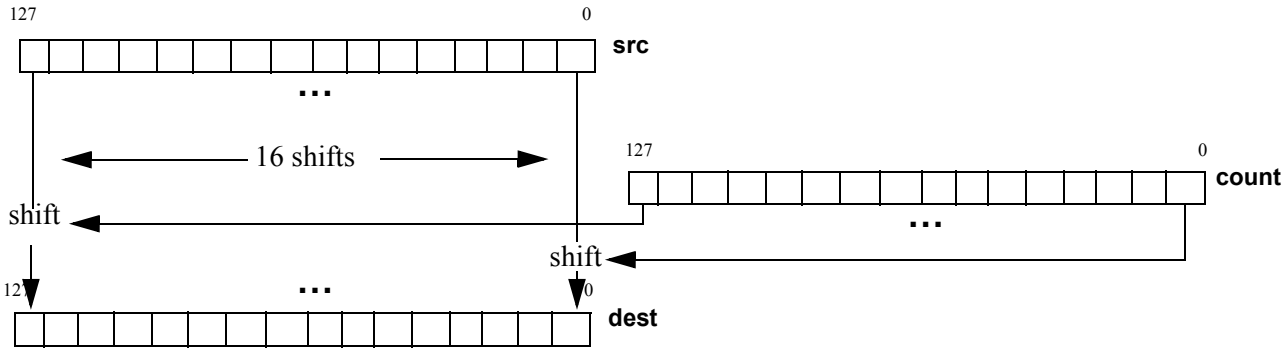
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHLB instruction requires three operands:

PSHLB dest, src, count

The PSHLB instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHLB *xmm1*, *xmm2, xmm3/mem128*	0F 24 44 /r /drex0	Shifts each byte of the second operand by an amount specified in the corresponding byte in the third operand and writes the result in the corresponding byte of the destination (xmm1 register).
PSHLB *xmm1*, *xmm3/mem128, xmm2*	0F 24 44 /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLW, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHLD Packed Shift Logical Doublewords

Shifts each of the four doublewords of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding doubleword of the destination.

The count byte is located in the low-order byte of the corresponding doubleword of the count operand.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the doubleword.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). Zeros are shifted in at the left end (most-significant bit) of the doubleword.

The shift amount is stored in two's-complement form. The count is modulo 32.

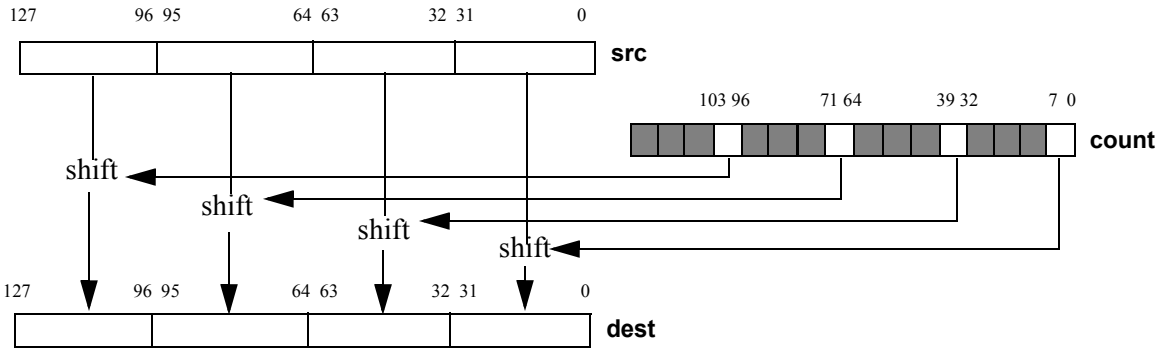
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHLD instruction requires three operands:

PSHLD dest, src, count

The PSHLD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHLD *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 46 /r /drex0	Shifts each doubleword of second operand by an amount specified in the low-order byte of the corresponding doubleword in the third operand and writes the result in the corresponding doubleword of the destination (xmm1 register).
PSHLD *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 46 /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.

Exception	Real	Virtual 8086	Protected	Cause of Exception
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 while MXCSR.MM=1.

PSHLQ Packed Shift Logical Quadwords

Shifts the two quadwords of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding quadword of the destination.

The count byte is located in the low-order byte of the corresponding quadword of the count operand. Bit 6 of the count byte is ignored.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the quadword.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). Zeros are shifted in at the left end (most-significant bit) of the quadword.

The shift amount is stored in two's-complement form. The count is modulo 64.

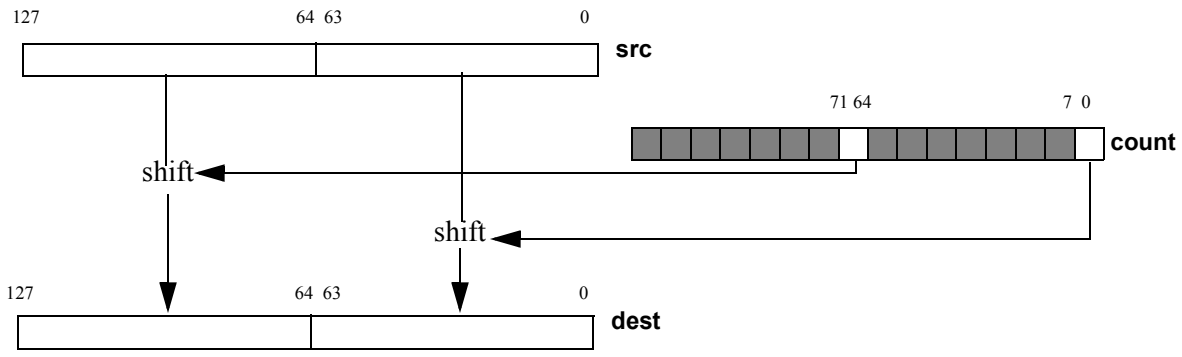
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHLQ instruction requires:

PSHLQ dest, src, count

The PSHLQ instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHLQ *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 47 /r /drex0	Shifts each quadword of second operand by an amount specified in the low-order byte of the corresponding quadword in the third operand and writes the result in the corresponding quadword of the destination (xmm1 register).
PSHLQ *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 47 /r /drex1	



Related Instructions

PROTB, PROTW, PROTD, PROTQ, PSHLB, PSHLW, PSHLD, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PSHLW Packed Shift Logical Words

Shifts each of the eight words of the source operand by the amount specified in the signed value of the corresponding *count* byte and writes the result in the corresponding word of the destination.

The count byte is located in the low-order byte of the corresponding word of the count operand.

If the count value is positive, bits are shifted to the left (toward the more significant bit positions). Zeros are shifted in at the right end (least-significant bit) of the word.

If the count value is negative, bits are shifted to the right (toward the least significant bit positions). Zeros are shifted in at the left end (most-significant bit) of the word.

The shift amount is stored in two's-complement form. The count is modulo 16.

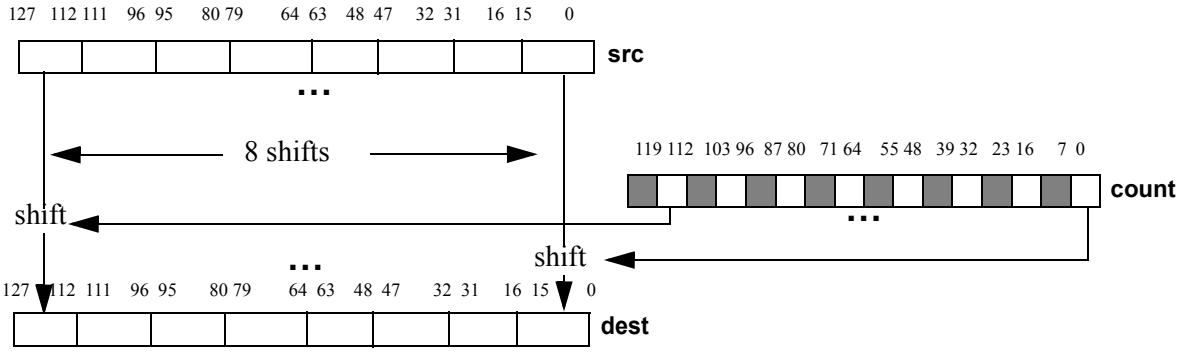
The first instruction operand is the destination register and is an XMM register addressed by the DREX.dest field.

The PSHLW instruction requires three operands:

PSHLW dest, src, count

The PSHLW instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PSHLW *xmm1*, *xmm2, xmm3*/*mem128*	0F 24 45 /r /drex0	Shifts each word of the second operand by an amount specified in the low-order byte of the corresponding word in the third operand and writes the result in the corresponding word of the destination (xmm1 register).
PSHLW *xmm1*, *xmm3*/*mem128, xmm2*	0F 24 45 /r /drex1	



Related Instructions

PROTB, PROLW, PROTD, PROTQ, PSHLB, PSHLD, PSHLQ, PSHAB, PSHAW, PSHAD, PSHAQ

rFLAGS Affected

None

MXCSR Flags Affected

None

Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.

PTEST Predicate Test Register

Performs a bitwise logical AND between the source XMM register or 128-bit memory location and destination XMM register. Sets the ZF flag to 1 if all bit positions specified in the mask operand are set to 0 in the source operand, and clears it otherwise. Sets the CF flag to 1 if all bit positions specified in the mask operand are set to 1 in the source operand. The first operand contains the source bits, the second operand contains the mask.

The PTEST instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
PTEST *xmm1*, *xmm2/mem128*	66 0F 38 17 /r	Set ZF, if the result of a logical AND of all bits in xmm2/m128 with the corresponding bits in xmm1 is 0s. Set CF, if the result of the logical AND of the source with a logical NOT of the destination is 0s.

Action

```
IF ((MASK[127:0] AND SRC[127:0]) = 0)
          ZF=1
      ELSE
          ZF=0

IF ((MASK[127:0] AND NOT SRC[127:0]) = 0)
          CF=1
      ELSE
          CF=0
```

Related Instructions

TEST

rFLAGS Affected

ID	VIP	VIF	AC	VM	RF	NT	IOPL	OF	DF	IF	TF	SF	ZF	AF	PF	CF
								0				0	M	0	0	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. Undefined flags are U.*

MXCSR Flags Affected

None

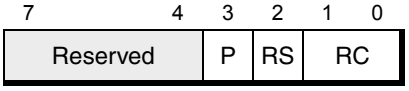
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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.

ROUNDPD Round Packed Double-Precision Floating-Point

Rounds each of the two double-precision floating-point values in an XMM register or a 128-bit memory location to the nearest integer, as determined by the rounding mode specified by the 8-bit immediate operand and writes the floating-point results in the corresponding 64 bits in a destination XMM register.

The 8-bit immediate operand specifies three control fields for the rounding operation.



Bits	Mnemonic	Description
7–4	Reserved	
3	P	Precision Mask
2	RS	Rounding Select
1–0	RC	Rounding Control

The precision mask (P) of the 8-bit immediate operand defines how the processor handles a precision exception. When the P bit is clear, the ROUNDPD instruction reports precision exceptions when an input is not an integer; when the P bit is set, ROUNDPD will not report precision exceptions.

The rounding select bit (RS) specifies the rounding mode control source. If the RS bit is set to 1, the rounding mode is determined by the value of the MXCSR.RC field; if the RS bit is cleared to zero, the rounding mode is determined by the RC field of the 8-bit immediate operand.

The rounding control (RC) field specifies a non-sticky rounding-mode value.

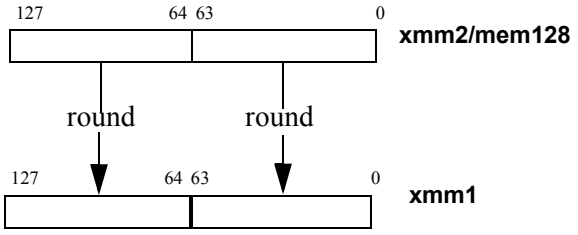
Table 2-4. Rounding Modes and Encoding of Rounding Control (RC) Field

Rounding Mode	RC Field Setting	Description
Round to Nearest (Even)	00b	The rounded result is that which is closest to the infinitely precise result. If two values are equally close, the integer value with the least-significant bit of zero (the even value) is returned.
Round Down (Toward –∞)	01b	The rounded result is closest to but no greater than the infinitely precise result.
Round Up (Toward +∞)	10b	The rounded result is closest to but no less than the infinitely precise result.
Round Toward Zero (Truncate)	11b	The rounded result is closest to the infinitely precise result but no greater in absolute value.

If any source operand is an SNaN, it will be converted to a QNaN. If DAZ is set to 1, then denormals are rounded to signed zero regardless of rounding mode.

The ROUNDPD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
ROUNDPD *xmm1*, *xmm2*/*mem128*,*imm8*	66 0F 3A 09 /r *ib*	Rounds two packed double-precision floating-point values in *xmm2* or 128-bit memory location and writes the results in the destination (*xmm1* register).



Related Instructions

ROUNDPS, ROUNDSD, ROUNDSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M					M
17	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.*

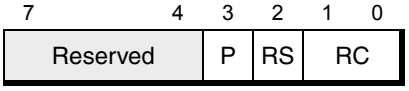
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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
Precision exception (PE)	X	X	X	The source operand was not an integral value.

ROUNDPS Round Packed Single-Precision Floating-Point

Rounds each of the four single-precision floating-point values in an XMM register or a 128-bit memory location to the nearest integer, as determined by the rounding mode specified by the 8-bit immediate operand and writes the floating-point results in the corresponding 32 bits in a destination XMM register.

The 8-bit immediate operand specifies three control fields for the rounding operation.



Bits	Mnemonic	Description
7–4	Reserved	
3	P	Precision Mask
2	RS	Rounding Select
1–0	RC	Rounding Control

The precision mask (P) of the 8-bit immediate operand defines how the processor handles a precision exception. When the P bit is clear, the ROUNDPS instruction reports precision exceptions when an input is not an integer; when the P bit is set, ROUNDPS will not report precision exceptions.

The rounding select bit (RS) specifies the rounding mode control source. If the RS bit is set to 1, the rounding mode is determined by the value of the MXCSR.RC field; if the RS bit is cleared to zero, the rounding mode is determined by then RC field of the 8-bit immediate operand.

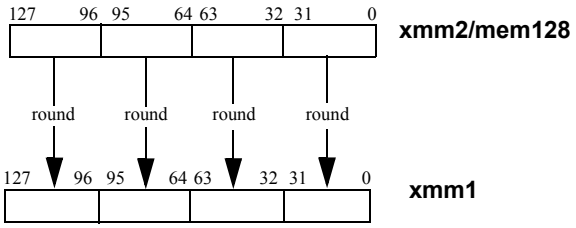
Table 2-5. Rounding Modes and Encoding of Rounding Control (RC) Field

Rounding Mode	RC Field Setting	Description
Round to Nearest (Even)	00b	The rounded result is that which is closest to the infinitely precise result. If two values are equally close, the integer value with the least-significant bit of zero (the even value) is returned.
Round Down (Toward –∞)	01b	The rounded result is closest to but no greater than the infinitely precise result.
Round Up (Toward +∞)	10b	The rounded result is closest to but no less than the infinitely precise result.
Round Toward Zero (Truncate)	11b	The rounded result is closest to the infinitely precise result but no greater in absolute value.

If any source operand is an SNaN, it will be converted to a QNaN. If DAZ is set to 1, then denormals are rounded to signed zero regardless of rounding mode.

The ROUNDPS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
ROUNDPS *xmm1*, *xmm2*/*mem128*,*imm8*	66 0F 3A 08 /r *ib*	Rounds four packed single-precision floating-point values in *xmm2* or 128-bit memory location and writes the results in the destination (*xmm1* register).



Related Instructions

ROUNDPD, ROUNDSD, ROUNDSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M					M
17	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.*

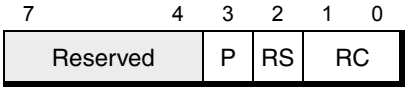
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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 while MXCSR.MM=0.
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 while MXCSR.MM=1.
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
Precision exception (PE)	X	X	X	The source operand was not an integral value.

ROUNDSD Round Scalar Double-Precision Floating-Point

Rounds the double-precision floating-point value in the low position of an XMM register or a 64-bit memory location to the nearest integer, as determined by the rounding mode specified by the 8-bit immediate operand and writes the results as a double-precision floating-point value in the low 64 bits of the destination XMM register. The upper double-precision floating-point value in the destination register is not affected.

The 8-bit immediate operand specifies three control fields for the rounding operation.



Bits	Mnemonic	Description
7–4	Reserved	
3	P	Precision Mask
2	RS	Rounding Select
1–0	RC	Rounding Control

The precision mask (P) of the 8-bit immediate operand defines how the processor handles a precision exception. When the P bit is clear, the ROUNDSD instruction reports precision exceptions when an input is not an integer; when the P bit is set, ROUNDSD will not report precision exceptions.

The rounding select bit (RS) specifies the rounding mode control source. If the RS bit is set to 1, the rounding mode is determined by the value of the MXCSR.RC field; if the RS bit is cleared to zero, the rounding mode is determined by then RC field of the 8-bit immediate operand.

The rounding control (RC) field specifies a non-sticky rounding-mode value.

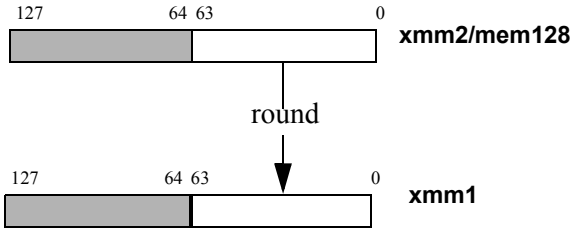
Table 2-6. Rounding Modes and Encoding of Rounding Control (RC) Field

Rounding Mode	RC Field Setting	Description
Round to Nearest (Even)	00b	The rounded result is that which is closest to the infinitely precise result. If two values are equally close, the integer value with the least-significant bit of zero (the even value) is returned.
Round Down (Toward –∞)	01b	The rounded result is closest to but no greater that the infinitely precise result.
Round Up (Toward +∞)	10b	The rounded result is closest to but no less than the infinitely precise result.
Round Toward Zero (Truncate)	11b	The rounded result is closest to the infinitely precise result but no greater in absolute value.

If any source operand is an SNaN, it will be converted to a QNaN. If DAZ is set to 1, then denormals are rounded to signed zero regardless of rounding mode.

The ROUNDSD instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
ROUNDSD *xmm1*, *xmm2*/*mem64*,*imm8*	66 0F 3A 0B /r *ib*	Rounds the scalar double-precision floating-point value in the lowest position in *xmm2* or 64-bit memory location and writes the results in the lowest position in the destination (*xmm1* register).



Related Instructions

ROUNDPD, ROUNDPS, ROUNDSS

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M					M
17	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.*

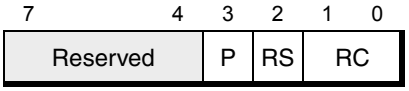
Exceptions

Exception	Real	Virtual 8086	Protected	Cause of Exception
Invalid opcode, #UD	X	X	X	The SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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
Precision exception (PE)	X	X	X	The source operand was not an integral value.

ROUNDSS Round Scalar Single-Precision Floating-Point

Rounds the single-precision floating-point value in the lowest position of an XMM register or a 32-bit memory location to the nearest integer, as determined by the rounding mode specified by the 8-bit immediate operand and writes the results as a double-precision floating-point value in the lowest 32 bits of the destination XMM register. The upper three single-precision floating-point values in the destination register are not affected.

The 8-bit immediate operand specifies three control fields for the rounding operation.



Bits	Mnemonic	Description
7–4	Reserved	
3	P	Precision Mask
2	RS	Rounding Select
1–0	RC	Rounding Control

The precision mask (P) of the 8-bit immediate operand defines how the processor handles a precision exception. When the P bit is clear, the ROUNDSS instruction reports precision exceptions when an input is not an integer; when the P bit is set, ROUNDSS will not report precision exceptions.

The rounding select bit (RS) specifies the rounding mode control source. If the RS bit is set to 1, the rounding mode is determined by the value of the MXCSR.RC field; if the RS bit is cleared to zero, the rounding mode is determined by then RC field of the 8-bit immediate operand.

The rounding control (RC) field specifies a non-sticky rounding-mode value.

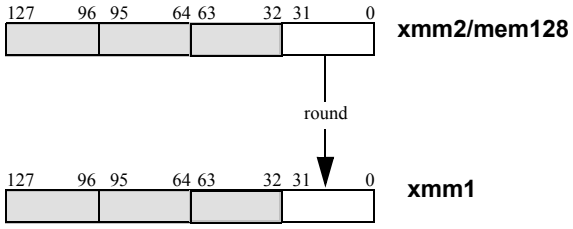
Table 2-7. Rounding Modes and Encoding of Rounding Control (RC) Field

Rounding Mode	RC Field Setting	Description
Round to Nearest (Even)	00b	The rounded result is that which is closest to the infinitely precise result. If two values are equally close, the integer value with the least-significant bit of zero (the even value) is returned.
Round Down (Toward –∞)	01b	The rounded result is closest to but no greater that the infinitely precise result.
Round Up (Toward +∞)	10b	The rounded result is closest to but no less than the infinitely precise result.
Round Toward Zero (Truncate)	11b	The rounded result is closest to the infinitely precise result but no greater in absolute value.

If any source operand is an SNaN, it will be converted to a QNaN. If DAZ is set to 1, then denormals are rounded to signed zero regardless of rounding mode.

The ROUNDSS instruction is an SSE5 instruction. The presence of this instruction set is indicated by a CPUID feature bit. (See the *CPUID Specification*, order# 25481.)

Mnemonic	Opcode	Description
ROUNDSS *xmm1*, *xmm2*/*mem32*, *imm8*	66 0F 3A 0A /r *ib*	Rounds the scalar single-precision floating-point value in the lowest position in *xmm2* or 32-bit memory location and writes the result in the lowest position in the destination (*xmm1* register).



Related Instructions

ROUNDPD, ROUNDPS, ROUNDSD

rFLAGS Affected

None

MXCSR Flags Affected

MM	FZ	RC		PM	UM	OM	ZM	DM	IM	DAZ	PE	UE	OE	ZE	DE	IE
											M					M
17	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 SSE5 instructions are not supported, as indicated by ECX bit 11 of CPUID function 8000_0001h.
	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
Precision exception (PE)	X	X	X	The source operand was not an integral value.