

# x86 Assembly Language Reference Manual

ORACLE®

Part No: E54851  
March 2018



**Part No: E54851**

Copyright © 1993, 2018, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

**Access to Oracle Support**

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

**Référence: E54851**

Copyright © 1993, 2018, Oracle et/ou ses affiliés. Tous droits réservés.

Ce logiciel et la documentation qui l'accompagne sont protégés par les lois sur la propriété intellectuelle. Ils sont concédés sous licence et soumis à des restrictions d'utilisation et de divulgation. Sauf stipulation expresse de votre contrat de licence ou de la loi, vous ne pouvez pas copier, reproduire, traduire, diffuser, modifier, accorder de licence, transmettre, distribuer, exposer, exécuter, publier ou afficher le logiciel, même partiellement, sous quelque forme et par quelque procédé que ce soit. Par ailleurs, il est interdit de procéder à toute ingénierie inverse du logiciel, de le désassembler ou de le décompiler, excepté à des fins d'interopérabilité avec des logiciels tiers ou tel que prescrit par la loi.

Les informations fournies dans ce document sont susceptibles de modification sans préavis. Par ailleurs, Oracle Corporation ne garantit pas qu'elles soient exemptes d'erreurs et vous invite, le cas échéant, à lui en faire part par écrit.

Si ce logiciel, ou la documentation qui l'accompagne, est livré sous licence au Gouvernement des Etats-Unis, ou à quiconque qui aurait souscrit la licence de ce logiciel pour le compte du Gouvernement des Etats-Unis, la notice suivante s'applique :

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

Ce logiciel ou matériel a été développé pour un usage général dans le cadre d'applications de gestion des informations. Ce logiciel ou matériel n'est pas conçu ni n'est destiné à être utilisé dans des applications à risque, notamment dans des applications pouvant causer un risque de dommages corporels. Si vous utilisez ce logiciel ou ce matériel dans le cadre d'applications dangereuses, il est de votre responsabilité de prendre toutes les mesures de secours, de sauvegarde, de redondance et autres mesures nécessaires à son utilisation dans des conditions optimales de sécurité. Oracle Corporation et ses affiliés déclinent toute responsabilité quant aux dommages causés par l'utilisation de ce logiciel ou matériel pour des applications dangereuses.

Oracle et Java sont des marques déposées d'Oracle Corporation et/ou de ses affiliés. Tout autre nom mentionné peut correspondre à des marques appartenant à d'autres propriétaires qu'Oracle.

Intel et Intel Xeon sont des marques ou des marques déposées d'Intel Corporation. Toutes les marques SPARC sont utilisées sous licence et sont des marques ou des marques déposées de SPARC International, Inc. AMD, Opteron, le logo AMD et le logo AMD Opteron sont des marques ou des marques déposées d'Advanced Micro Devices. UNIX est une marque déposée de The Open Group.

Ce logiciel ou matériel et la documentation qui l'accompagne peuvent fournir des informations ou des liens donnant accès à des contenus, des produits et des services émanant de tiers. Oracle Corporation et ses affiliés déclinent toute responsabilité ou garantie expresse quant aux contenus, produits ou services émanant de tiers, sauf mention contraire stipulée dans un contrat entre vous et Oracle. En aucun cas, Oracle Corporation et ses affiliés ne sauraient être tenus pour responsables des pertes subies, des coûts occasionnés ou des dommages causés par l'accès à des contenus, produits ou services tiers, ou à leur utilisation, sauf mention contraire stipulée dans un contrat entre vous et Oracle.

**Accès aux services de support Oracle**

Les clients Oracle qui ont souscrit un contrat de support ont accès au support électronique via My Oracle Support. Pour plus d'informations, visitez le site <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info> ou le site <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> si vous êtes malentendant.

# Contents

---

<b>Using This Documentation</b> .....	13
<b>1 Overview of the Oracle Solaris x86 Assembler</b> .....	15
1.1 Assembler Overview .....	15
1.2 Syntax Differences Between x86 Assemblers .....	15
<b>2 Oracle Solaris x86 Assembly Language Syntax</b> .....	17
2.1 Assembly Language Lexical Conventions .....	17
2.1.1 Assembly Language Statements .....	17
2.1.2 Assembly Language Tokens .....	19
2.2 Assembly Language Instructions, Operands, and Addressing .....	21
2.2.1 Assembly Language Instructions .....	22
2.2.2 Assembly Language Operands .....	23
2.3 Assembler Directives .....	24
<b>3 Instruction Set Mapping</b> .....	33
3.1 Instruction Overview .....	34
3.2 General-Purpose Instructions .....	35
3.2.1 Data Transfer Instructions .....	36
3.2.2 Binary Arithmetic Instructions .....	39
3.2.3 Decimal Arithmetic Instructions .....	39
3.2.4 Logical Instructions .....	40
3.2.5 Shift and Rotate Instructions .....	40
3.2.6 Bit and Byte Instructions .....	41
3.2.7 Control Transfer Instructions .....	42
3.2.8 String Instructions .....	44
3.2.9 I/O Instructions .....	45
3.2.10 Flag Control (EFLAG) Instructions .....	45

3.2.11	Segment Register Instructions .....	46
3.2.12	Miscellaneous Instructions .....	46
3.3	Floating-Point Instructions .....	47
3.3.1	Data Transfer Instructions (Floating Point) .....	47
3.3.2	Basic Arithmetic Instructions (Floating-Point) .....	48
3.3.3	Comparison Instructions (Floating-Point) .....	49
3.3.4	Transcendental Instructions (Floating-Point) .....	50
3.3.5	Load Constants (Floating-Point) Instructions .....	50
3.3.6	Control Instructions (Floating-Point) .....	51
3.4	SIMD State Management Instructions .....	52
3.5	ADX Instructions .....	53
3.6	AES Instructions .....	53
3.6.1	Advanced Vector Extensions of AES Instructions .....	53
3.7	AVX Instructions .....	54
3.8	AVX2 Instructions .....	77
3.9	BMI1 Instructions .....	85
3.10	BMI2 Instructions .....	85
3.11	F16C Instructions .....	86
3.12	FMA Instructions .....	86
3.13	FSGSBASE Instructions .....	92
3.14	MMX Instructions .....	93
3.14.1	Data Transfer Instructions (MMX) .....	93
3.14.2	Conversion Instructions (MMX) .....	93
3.14.3	Packed Arithmetic Instructions (MMX) .....	94
3.14.4	Comparison Instructions (MMX) .....	95
3.14.5	Logical Instructions (MMX) .....	96
3.14.6	Shift and Rotate Instructions (MMX) .....	96
3.14.7	State Management Instructions (MMX) .....	97
3.15	MOVBE Instructions .....	97
3.16	PCLMULQDQ Instructions .....	97
3.17	PREFETCH Instructions .....	98
3.18	RDRAND Instructions .....	98
3.19	RDSEED Instructions .....	98
3.20	SSE Instructions .....	98
3.20.1	SIMD Single-Precision Floating-Point Instructions (SSE) .....	99
3.20.2	MXCSR State Management Instructions (SSE) .....	104
3.20.3	64-Bit SIMD Integer Instructions (SSE) .....	104

3.20.4	Miscellaneous Instructions (SSE)	105
3.21	SSE2 Instructions	106
3.21.1	SSE2 Packed and Scalar Double-Precision Floating-Point Instructions	106
3.21.2	SSE2 Packed Single-Precision Floating-Point Instructions	111
3.21.3	SSE2 128-Bit SIMD Integer Instructions	112
3.21.4	SSE2 Miscellaneous Instructions	112
3.22	SSE3 Instructions	113
3.23	SSE4a Instructions	114
3.24	SSE4.1 Instructions	114
3.25	SSE4.2 Instructions	117
3.26	SSSE3 Instructions	117
3.27	Transactional Synchronization Extensions	118
3.28	Operating System Support Instructions	119
3.29	VMX Instructions	120
3.30	XSAVE Instructions	121
3.31	3DNow Instructions	121
3.32	64-Bit AMD Opteron Considerations	123
<b>A</b>	<b>Using the Assembler Command Line</b>	125
A.1	Assembler Command Line	125
A.2	Assembler Command Line Options	126
A.3	Disassembling Object Code	129
<b>Index</b>		131





## Tables

---

<b>TABLE 1</b>	Instruction References .....	34
<b>TABLE 2</b>	Data Transfer Instructions .....	36
<b>TABLE 3</b>	Binary Arithmetic Instructions .....	39
<b>TABLE 4</b>	Decimal Arithmetic Instructions .....	39
<b>TABLE 5</b>	Logical Instructions .....	40
<b>TABLE 6</b>	Shift and Rotate Instructions .....	40
<b>TABLE 7</b>	Bit and Byte Instructions .....	41
<b>TABLE 8</b>	Control Transfer Instructions .....	42
<b>TABLE 9</b>	String Instructions .....	44
<b>TABLE 10</b>	I/O Instructions .....	45
<b>TABLE 11</b>	Flag Control Instructions .....	46
<b>TABLE 12</b>	Segment Register Instructions .....	46
<b>TABLE 13</b>	Miscellaneous Instructions .....	47
<b>TABLE 14</b>	Data Transfer Instructions (Floating-Point) .....	47
<b>TABLE 15</b>	Basic Arithmetic Instructions (Floating-Point) .....	48
<b>TABLE 16</b>	Comparison Instructions (Floating-Point) .....	49
<b>TABLE 17</b>	Transcendental Instructions (Floating-Point) .....	50
<b>TABLE 18</b>	Load Constants Instructions (Floating-Point) .....	50
<b>TABLE 19</b>	Control Instructions (Floating-Point) .....	51
<b>TABLE 20</b>	SIMD State Management Instructions .....	52
<b>TABLE 21</b>	ADX Instructions .....	53
<b>TABLE 22</b>	AES Instructions .....	53
<b>TABLE 23</b>	Advanced Vector Extensions of AES Instructions .....	53
<b>TABLE 24</b>	AVX Instructions .....	54
<b>TABLE 25</b>	AVX2 Instructions .....	77
<b>TABLE 26</b>	BMI1 Instructions .....	85
<b>TABLE 27</b>	BMI2 Instructions .....	85
<b>TABLE 28</b>	F16C Instructions .....	86

---

TABLE 29	FMA Instructions .....	86
TABLE 30	FSGSBASE Instructions .....	92
TABLE 31	Data Transfer Instructions (MMX) .....	93
TABLE 32	Conversion Instructions (MMX) .....	93
TABLE 33	Packed Arithmetic Instructions (MMX) .....	94
TABLE 34	Comparison Instructions (MMX) .....	95
TABLE 35	Logical Instructions (MMX) .....	96
TABLE 36	Shift and Rotate Instructions (MMX) .....	96
TABLE 37	State Management Instructions (MMX) .....	97
TABLE 38	MOVBE Instructions .....	97
TABLE 39	PCLMULQDQ Instructions .....	97
TABLE 40	PREFETCH Instructions .....	98
TABLE 41	RDRAND Instructions .....	98
TABLE 42	RDSEED Instructions .....	98
TABLE 43	Data Transfer Instructions (SSE) .....	99
TABLE 44	Packed Arithmetic Instructions (SSE) .....	100
TABLE 45	Comparison Instructions (SSE) .....	102
TABLE 46	Logical Instructions (SSE) .....	102
TABLE 47	Shuffle and Unpack Instructions (SSE) .....	103
TABLE 48	Conversion Instructions (SSE) .....	103
TABLE 49	MXCSR State Management Instructions (SSE) .....	104
TABLE 50	64-Bit SIMD Integer Instructions (SSE) .....	104
TABLE 51	Miscellaneous Instructions (SSE) .....	105
TABLE 52	SSE2 Data Movement Instructions .....	106
TABLE 53	SSE2 Packed Arithmetic Instructions .....	107
TABLE 54	SSE2 Logical Instructions .....	108
TABLE 55	SSE2 Compare Instructions .....	109
TABLE 56	SSE2 Shuffle and Unpack Instructions .....	109
TABLE 57	SSE2 Conversion Instructions .....	110
TABLE 58	SSE2 Packed Single-Precision Floating-Point Instructions .....	111
TABLE 59	SSE2 128-Bit SIMD Integer Instructions .....	112
TABLE 60	SSE2 Miscellaneous Instructions .....	113
TABLE 61	SSE3 Instructions .....	113
TABLE 62	SSE4a Instructions .....	114
TABLE 63	SSE4.1 Instructions .....	114
TABLE 64	SSE4.2 Instructions .....	117

---

<b>TABLE 65</b>	SSSE3 Instructions .....	117
<b>TABLE 66</b>	HLE Instructions .....	118
<b>TABLE 67</b>	RTM Instructions .....	118
<b>TABLE 68</b>	Operating System Support Instructions .....	119
<b>TABLE 69</b>	VMX Instructions .....	120
<b>TABLE 70</b>	XSAVE Instructions .....	121
<b>TABLE 71</b>	3DNow Instructions .....	121



## Using This Documentation

---

- **Overview** – Provides information that helps experienced assembly language programmers understand disassembled output of Oracle Solaris compilers

This manual documents the syntax of the Oracle Solaris x86 assembly language. This manual is neither an introductory book about assembly language programming nor a reference manual for the x86 architecture.

- **Audience** – This manual is intended for experienced x86 assembly language programmers who are familiar with the x86 architecture.
- **Required knowledge** – You should have a thorough knowledge of assembly language programming in general and be familiar with the x86 architecture in specific. You should be familiar with the ELF object file format.

## Product Documentation Library

Documentation and resources for this product and related products are available at <http://www.oracle.com/pls/topic/lookup?ctx=E53394-01>.

## Feedback

Provide feedback about this documentation at <http://www.oracle.com/goto/docfeedback>.



# Overview of the Oracle Solaris x86 Assembler

---

This chapter provides a brief overview of the Oracle Solaris x86 assembler as. This chapter discusses the following topics:

- [“1.1 Assembler Overview” on page 15](#)
- [“1.2 Syntax Differences Between x86 Assemblers” on page 15](#)

## 1.1 Assembler Overview

The Oracle Solaris x86 assembler as translates Oracle Solaris x86 assembly language into Executable and Linking Format (ELF) relocatable object files that can be linked with other object files to create an executable file or a shared object file. (See [Chapter 14, “Object File Format”](#) in *Oracle Solaris 11.3 Linkers and Libraries Guide* for a complete discussion of ELF object file format.) The assembler supports macro processing by the C preprocessor (cpp) or the m4 macro processor.

## 1.2 Syntax Differences Between x86 Assemblers

There is no standard assembly language for the x86 architecture. Vendor implementations of assemblers for the x86 architecture instruction sets differ in syntax and functionality. The syntax of the Oracle Solaris x86 assembler is compatible with the syntax of the assembler distributed with earlier releases of the UNIX operating system (this syntax is sometimes termed “AT&T syntax”). Developers familiar with other assemblers derived from the original UNIX assemblers, such as the Free Software Foundation's *gas*, will find the syntax of the Oracle Solaris x86 assembler very straightforward.

However, the syntax of x86 assemblers distributed by Intel and Microsoft (sometimes termed “Intel syntax”) differs significantly from the syntax of the Oracle Solaris x86 assembler. These differences are most pronounced in the handling of instruction operands:

- The Oracle Solaris and Intel assemblers use the opposite order for source and destination operands.
- The Oracle Solaris assembler specifies the size of memory operands by adding a suffix to the instruction mnemonic, while the Intel assembler prefixes the memory operands.
- The Oracle Solaris assembler prefixes immediate operands with a dollar sign (\$) (ASCII 0x24), while the Intel assembler does not delimit immediate operands.

See [Chapter 2, “Oracle Solaris x86 Assembly Language Syntax”](#) for additional differences between x86 assemblers.



# Oracle Solaris x86 Assembly Language Syntax

---

This chapter documents the syntax of the Oracle Solaris x86 assembly language.

- [“2.1 Assembly Language Lexical Conventions” on page 17](#)
- [“2.2 Assembly Language Instructions, Operands, and Addressing” on page 21](#)
- [“2.3 Assembler Directives” on page 24](#)

## 2.1 Assembly Language Lexical Conventions

This section discusses the lexical conventions of the Oracle Solaris x86 assembly language.

### 2.1.1 Assembly Language Statements

An x86 assembly language program consists of one or more files containing *statements*. A *statement* consists of *tokens* separated by *whitespace* and terminated by either a newline character (ASCII 0x0A) or a semicolon (;) (ASCII 0x3B). *Whitespace* consists of spaces (ASCII 0x20), tabs (ASCII 0x09), and form feeds (ASCII 0x0B) that are not contained in a string or comment. More than one statement can be placed on a single input line provided that each statement is terminated by a semicolon. A statement can consist of a *comment*. *Empty statements*, consisting only of whitespace, are allowed.

#### 2.1.1.1 Assembly Language Comments

A *comment* can be appended to a statement. The comment consists of the slash character (/) (ASCII 0x2F) followed by the text of the comment. The comment is terminated by the newline that terminates the statement.

### 2.1.1.2 Assembly Language Labels

A *label* can be placed at the beginning of a statement. During assembly, the label is assigned the current value of the active location counter and serves as an instruction operand. There are two types of labels: *symbolic* and *numeric*.

#### Assembly Language Symbolic Labels

A *symbolic* label consists of an *identifier* (or *symbol*) followed by a colon (:) (ASCII 0x3A). Symbolic labels must be defined only once. Symbolic labels have *global* scope and appear in the object file's symbol table.

Symbolic labels with identifiers beginning with a period (.) (ASCII 0x2E) are considered to have *local* scope and are not included in the object file's symbol table.

#### Assembly Language Numeric Labels

A *numeric* label consists of a unsigned decimal *int32* value followed by a colon (:). Numeric labels are used only for local reference and are not included in the object file's symbol table. Numeric labels have limited scope and can be redefined repeatedly.

When a numeric label is used as a reference (as an instruction operand, for example), the suffixes *b* ("backward") or *f* ("forward") should be added to the numeric label. For numeric label *N*, the reference *Nb* refers to the nearest label *N* defined *before* the reference, and the reference *Nf* refers to the nearest label *N* defined *after* the reference. The following example illustrates the use of numeric labels:

```
1:          / define numeric label "1"
one:       / define symbolic label "one"

/ ... assembler code ...

jmp 1f     / jump to first numeric label "1" defined
          / after this instruction
          / (this reference is equivalent to label "two")

jmp 1b     / jump to last numeric label "1" defined
          / before this instruction
          / (this reference is equivalent to label "one")

1:        / redefine label "1"
```

```
two:          / define symbolic label "two"

jmp  1b      / jump to last numeric label "1" defined
          / before this instruction
          / (this reference is equivalent to label "two")
```

## 2.1.2 Assembly Language Tokens

There are five classes of tokens:

- Identifiers (symbols)
- Keywords
- Numerical constants
- String Constants
- Operators

### 2.1.2.1 Assembly Language Identifiers

An *identifier* is an arbitrarily-long sequence of letters and digits. The first character must be a letter; the underscore (`_`) (ASCII 0x5F) and the period (`.`) (ASCII 0x2E) are considered to be letters. Case is significant: uppercase and lowercase letters are different.

### 2.1.2.2 Assembly Language Keywords

*Keywords* such as x86 instruction mnemonics ("opcodes") and assembler directives are reserved for the assembler and should not be used as identifiers. See [Chapter 3, "Instruction Set Mapping"](#) for a list of the Oracle Solaris x86 mnemonics. See ["2.3 Assembler Directives" on page 24](#) for the list of as assembler directives.

### 2.1.2.3 Numerical Constants

Numbers in the x86 architecture can be *integers* or *floating point*. Integers can be *signed* or *unsigned*, with signed integers represented in two's complement representation. Floating-point numbers can be: single-precision floating-point; double-precision floating-point; and double-extended precision floating-point.

## Integer Constants

*Integers* can be expressed in several bases:

- **Decimal.** Decimal integers begin with a non-zero digit followed by zero or more decimal digits (0-9).
- **Binary.** Binary integers begin with "0b" or "0B" followed by zero or more binary digits (0, 1).
- **Octal.** Octal integers begin with zero (0) followed by zero or more octal digits (0-7).
- **Hexadecimal.** Hexadecimal integers begin with "0x" or "0X" followed by one or more hexadecimal digits (0-9, A–F). Hexadecimal digits can be either uppercase or lowercase.

## Assembly Language Floating Point Constants

*Floating point* constants have the following format:

- **Sign** (optional) – Either plus (+) or minus (–)
- **Integer** (optional) – Zero or more decimal digits (0–9)
- **Fraction** (optional) – Decimal point (.) followed by zero or more decimal digits
- **Exponent** (optional) – The letter "e" or "E", followed by an optional sign (plus or minus), followed by one or more decimal digits (0-9)

A valid floating point constant must have either an integer part or a fractional part.

### 2.1.2.4 Assembly Language String Constants

A *string* constant consists of a sequence of characters enclosed in double quotes (") (ASCII 0x22). To include a double-quote character ("), single-quote character ('), or backslash character (\) within a string, precede the character with a backslash (\) (ASCII 0x5C). A character can be expressed in a string as its ASCII value in octal preceded by a backslash (for example, the letter "J" could be expressed as "\112"). The assembler accepts the following escape sequences in strings:

Escape Sequence	Character Name	ASCII Value (hex)
\n	newline	0A
\r	carriage return	0D
\b	backspace	08
\t	horizontal tab	09

Escape Sequence	Character Name	ASCII Value (hex)
\f	form feed	0C
\v	vertical tab	0B

### 2.1.2.5 Assembly Language Operators

The assembler supports the following operators for use in expressions. Operators have no assigned precedence. Expressions can be grouped in square brackets ( [ ] ) to establish precedence.

+	Addition
-	Subtraction
\*	Multiplication
∕	Division
&	Bitwise logical AND
	Bitwise logical OR
>>	Shift right
<<	Shift left
\%	Remainder
!	Bitwise logical AND NOT
^	Bitwise logical XOR

---

**Note** - The asterisk (\*), slash (/), and percent sign (%) characters are overloaded. When used as operators in an expression, these characters must be preceded by the backslash character (\).

---

## 2.2 Assembly Language Instructions, Operands, and Addressing

*Instructions* are operations performed by the CPU. *Operands* are entities operated upon by the instruction. *Addresses* are the locations in memory of specified data.

## 2.2.1 Assembly Language Instructions

An *instruction* is a statement that is executed at runtime. An x86 instruction statement can consist of four parts:

- Label (optional)
- Instruction (required)
- Operands (instruction specific)
- Comment (optional)

See [“2.1.1 Assembly Language Statements” on page 17](#) for the description of labels and comments.

The terms *instruction* and *mnemonic* are used interchangeably in this document to refer to the names of x86 instructions. Although the term *opcode* is sometimes used as a synonym for *instruction*, this document reserves the term *opcode* for the hexadecimal representation of the instruction value.

For most instructions, the Oracle Solaris x86 assembler mnemonics are the same as the Intel or AMD mnemonics. However, the Oracle Solaris x86 mnemonics might appear to be different because the Oracle Solaris mnemonics are suffixed with a one-character modifier that specifies the size of the instruction operands. That is, the Oracle Solaris assembler derives its operand type information from the instruction name and the suffix. If a mnemonic is specified with no type suffix, the operand type defaults to long. Possible operand types and their instruction suffixes are:

b	Byte (8-bit)
w	Word (16-bit)
l	Long (32-bit) (default)
q	Quadword (64-bit)

The assembler recognizes the following suffixes for x87 floating-point instructions:

[no suffix]	Instruction operands are registers only
l ("long")	Instruction operands are 64-bit
s ("short")	Instruction operands are 32-bit

See [Chapter 3, “Instruction Set Mapping”](#) for a mapping between Oracle Solaris x86 assembly language mnemonics and the equivalent Intel or AMD mnemonics.

## 2.2.2 Assembly Language Operands

An x86 instruction can have zero to three operands. Operands are separated by commas (,) (ASCII 0x2C). For instructions with two operands, the first (lefthand) operand is the *source* operand, and the second (righthand) operand is the *destination* operand (that is, *source* → *destination*).

---

**Note** - The Intel assembler uses the opposite order (*destination* ← *source*) for operands.

---

Operands can be *immediate* (that is, constant expressions that evaluate to an inline value), *register* (a value in the processor number registers), or *memory* (a value stored in memory). An *indirect* operand contains the address of the actual operand value. Indirect operands are specified by prefixing the operand with an asterisk (\*) (ASCII 0x2A). Only jump and call instructions can use indirect operands.

- *Immediate* operands are prefixed with a dollar sign (\$) (ASCII 0x24)
- *Register* names are prefixed with a percent sign (%) (ASCII 0x25)
- *Memory* operands are specified either by the name of a variable or by a register that contains the address of a variable. A variable name implies the address of a variable and instructs the computer to reference the contents of memory at that address. Memory references have the following syntax:  
*segment:offset(base, index, scale)*.
  - *Segment* is any of the x86 architecture segment registers. *Segment* is optional: if specified, it must be separated from *offset* by a colon (:). If *segment* is omitted, the value of %ds (the default segment register) is assumed.
  - *Offset* is the displacement from *segment* of the desired memory value. *Offset* is optional.
  - *Base* and *index* can be any of the general 32-bit number registers.
  - *Scale* is a factor by which *index* is to be multiplied before being added to *base* to specify the address of the operand. *Scale* can have the value of 1, 2, 4, or 8. If *scale* is not specified, the default value is 1.

Some examples of memory addresses are:

```
movl var, %eax
```

Move the contents of memory location `var` into number register `%eax`.

```
movl %cs:var, %eax
```

Move the contents of memory location `var` in the code segment (register `%cs`) into number register `%eax`.

```
movl $var, %eax
```

Move the address of `var` into number register `%eax`.

```
movl array_base(%esi), %eax
```

Add the address of memory location `array_base` to the contents of number register `%esi` to determine an address in memory. Move the contents of this address into number register `%eax`.

```
movl (%ebx, %esi, 4), %eax
```

Multiply the contents of number register `%esi` by 4 and add the result to the contents of number register `%ebx` to produce a memory reference. Move the contents of this memory location into number register `%eax`.

```
movl struct_base(%ebx, %esi, 4), %eax
```

Multiply the contents of number register `%esi` by 4, add the result to the contents of number register `%ebx`, and add the result to the address of `struct_base` to produce an address. Move the contents of this address into number register `%eax`.

## 2.3 Assembler Directives

*Directives* are commands that are part of the assembler syntax but are not related to the x86 processor instruction set. All assembler directives begin with a period (`.`) (ASCII 0x2E).

```
.align integer, pad
```

The `.align` directive causes the next data generated to be aligned modulo *integer* bytes. *Integer* must be a positive integer expression and must be a power of 2. If specified, *pad* is an integer byte value used for padding. The default value of *pad* for the text section is 0x90 (nop); for other sections, the default value of *pad* is zero (0).

```
.ascii "string"
```

The `.ascii` directive places the characters in *string* into the object module at the current location but does *not* terminate the string with a null byte (`\0`). *String* must be enclosed in double quotes (`"`) (ASCII 0x22). The `.ascii` directive is not valid for the `.bss` section.

```
.bcd integer
```

The `.bcd` directive generates a packed decimal (80-bit) value into the current section. The `.bcd` directive is not valid for the `.bss` section.



`.bss`

The `.bss` directive changes the current section to `.bss`.

`.bss symbol, integer`

Define *symbol* in the `.bss` section and add *integer* bytes to the value of the location counter for `.bss`. When issued with arguments, the `.bss` directive does not change the current section to `.bss`. *Integer* must be positive.

`.byte byte1, byte2, ..., byteN`

The `.byte` directive generates initialized bytes into the current section. The `.byte` directive is not valid for the `.bss` section. Each *byte* must be an 8-bit value.

`.2byte expression1, expression2, ..., expressionN`

Refer to the description of the `.value` directive.

`.4byte expression1, expression2, ..., expressionN`

Refer to the description of the `.long` directive.

`.8byte expression1, expression2, ..., expressionN`

Refer to the description of the `.quad` directive.

`.cfi_adjust_cfa_offset OFFSET`

The `.cfi_adjust_cfa_offset` directive is similar to `.cfi_def_cfa_offset` directive but *OFFSET* is a relative value that is added or subtracted from the previous offset.

`.cfi_def_cfa_offset OFFSET`

The `.cfi_def_cfa_offset` directive, modifies the rule for computing CFA. The value of the register remains the same, but *OFFSET* is new. Note that this is the absolute offset that will be added to a defined register to compute the CFA address.

`.cfi_def_cfa REGISTER, OFFSET`

The `.cfi_def_cfa` directive, defines a rule to compute CFA. This directive takes address from *REGISTER* and adds *OFFSET* to it.

`.cfi_def_cfa_register REGISTER`

The `.cfi_def_cfa_register` directive, modifies the rule for computing CFA. The register in the CFA is set to a new value. The offset remains the same.

`.cfi_endproc`

The `.cfi_endproc` directive, is used at the end of a function where it closes its unwind entry previously opened by `.cfi_startproc` and emits it to `.eh_frame`.

`.cfi_escape` *EXPRESSION* [, ...]

The `.cfi_escape` directive, allows you to add arbitrary bytes to the unwind information. You can use this directive to add OS-specific CFI opcodes, or generic CFI opcodes that the assembler does not support.

`.cfi_lsda` *encoding* [, *exp*]

The `.cfi_lsda` directive, defines LSDA and its encoding. The *encoding* should be a constant which determines how the LSDA should be encoded. If the value of *encoding* is 255 (`DW_EH_PE_omit`), second argument is not present, otherwise second argument should be a constant or a symbol name. The default directive used after `.cfi_startproc` directive is `.cfi_lsda 0xff`.

`.cfi_offset` *REGISTER*, *OFFSET*

The `.cfi_offset` directive, saves the previous value of *REGISTER* at offset *OFFSET* from CFA.

`.cfi_personality` *encoding* [, *exp*]

The `.cfi_personality` directive, defines the personality routine and its encoding. The *encoding* must be a constant which determines how the personality should be encoded. If the value of *encoding* is 255 (`DW_EH_PE_omit`), second argument is not present, otherwise second argument should be a constant or a symbol name. When you are using indirect encodings, the symbol provided should be the location where personality can be loaded from and not the personality routine itself. The default directive used after `.cfi_startproc` directive is `.cfi_personality 0xff`.

`.cfi_register` *REGISTER1 REGISTER2*

The `.cfi_register` *REGISTER1 REGISTER2* directive, saves the previous value of *REGISTER1* in register *REGISTER2*.

`.cfi_rel_offset` *REGISTER*, *OFFSET*

In the `.cfi_rel_offset` directive, saves the previous value of *REGISTER* at offset *OFFSET* from the current CFA register. This is transformed to `.cfi_offset` using the known displacement of the CFA register from the CFA. This is often easier to use, because the number will match the code it is annotating.

`.cfi_remember_state`

The `.cfi_remember_state` directive, saves all the current rules for all the registers. If the following `.cfi_*` directives is bad, then you can use the `.cfi_restore_state` directive to restore the previous saved state.

**.cfi\_restore REGISTER**

The `.cfi_restore` directive, indicates that the rule for register is now the same as it was at the beginning of the function, after all initial instructions added by `.cfi_startproc` directive are executed.

**.cfi\_restore\_state**

The `.cfi_restore_state` directive, restores the previous saved state of the register.

**.cfi\_return\_column REGISTER**

The `.cfi_return_column` directive, changes return column *REGISTER*. The return address is either directly in *REGISTER* or can be accessed by rules for *REGISTER*.

**.cfi\_same\_value REGISTER**

The `.cfi_same_value` directive, indicates the current value of *REGISTER* is the same like in the previous frame and does not require restoration.

**.cfi\_sections section\_list**

The `.cfi_sections section_list` directive, specifies if CFI directives should emit `.eh_frame` section and/or `.debug_frame` section. You can use `.eh_frame` as the *section\_list* to emit `.eh_frame`. You can use the `.debug_frame` as the *section\_list* to emit `.debug_frame`. To emit both use `.eh_frame` and `.debug_frame` as the *section\_list*. By default, `.cfi_sections` emits `.eh_frame`.

**.cfi\_startproc**

The `.cfi_startproc` directive, is used at the beginning of each function that should have an entry in `.eh_frame`. It initializes some internal data structures and emits architecture dependent initial CFI instructions. Each `.cfi_startproc` directive has to be closed by `.cfi_endproc`.

**.cfi\_undefined REGISTER**

The `.cfi_undefined` directive, indicates the point from which the previous value of the register cannot be restored.

**.comm name, size, alignment**

The `.comm` directive allocates storage in the data section. The storage is referenced by the identifier *name*. *Size* is measured in bytes and must be a positive integer. *Name* cannot be predefined. *Alignment* is optional. If *alignment* is specified, the address of *name* is aligned to a multiple of *alignment*.

**.data**

The `.data` directive changes the current section to `.data`.

**.double** *float*

The `.double` directive generates a double-precision floating-point constant into the current section. The `.double` directive is not valid for the `.bss` section.

**.even**

The `.even` directive aligns the current program counter (.) to an even boundary.

**.ext** *expression1, expression2, ..., expressionN*

The `.ext` directive generates an 80387 80-bit floating point constant for each *expression* into the current section. The `.ext` directive is not valid for the `.bss` section.

**.file** "*string*"

The `.file` directive creates a symbol table entry where *string* is the symbol name and `STT_FILE` is the symbol table type. *String* specifies the name of the source file associated with the object file.

**.float** *float*

The `.float` directive generates a single-precision floating-point constant into the current section. The `.float` directive is not valid in the `.bss` section.

**.globl** *symbol1, symbol2, ..., symbolN*

The `.globl` directive declares each *symbol* in the list to be *global*. Each symbol is either defined externally or defined in the input file and accessible in other files. Default bindings for the symbol are overridden. A global symbol definition in one file satisfies an undefined reference to the same global symbol in another file. Multiple definitions of a defined global symbol are not allowed. If a defined global symbol has more than one definition, an error occurs. The `.globl` directive only declares the symbol to be global in scope, it does not define the symbol.

**.group** *group, section, #comdat*

The `.group` directive adds *section* to a COMDAT *group*. Refer to [“COMDAT Section” in Oracle Solaris 11.3 Linkers and Libraries Guide](#) for additional information about COMDAT.

**.hidden** *symbol1, symbol2, ..., symbolN*

The `.hidden` directive declares each *symbol* in the list to have *hidden* linker scoping. All references to *symbol* within a dynamic module bind to the definition within that module. *Symbol* is not visible outside of the module.

**.ident** "*string*"

The `.ident` directive creates an entry in the `.comment` section containing *string*. *String* is any sequence of characters, not including the double quote ("). To include the double quote

character within a string, precede the double quote character with a backslash (\) (ASCII 0x5C).

`.lcomm name, size, alignment`

The `.lcomm` directive allocates storage in the `.bss` section. The storage is referenced by the symbol *name*, and has a size of *size* bytes. *Name* cannot be predefined, and *size* must be a positive integer. If *alignment* is specified, the address of *name* is aligned to a multiple of *alignment* bytes. If *alignment* is not specified, the default alignment is 4 bytes.

`.local symbol1, symbol2, ..., symbolN`

The `.local` directive declares each *symbol* in the list to be *local*. Each symbol is defined in the input file and not accessible to other files. Default bindings for the symbols are overridden. Symbols declared with the `.local` directive take precedence over *weak* and *global* symbols. (See “[Symbol Table Section](#)” in *Oracle Solaris 11.3 Linkers and Libraries Guide* for a description of global and weak symbols.) Because local symbols are not accessible to other files, local symbols of the same name may exist in multiple files. The `.local` directive only declares the symbol to be local in scope, it does not define the symbol.

`.long expression1, expression2, ..., expressionN`

The `.long` directive generates a long integer (32-bit, two's complement value) for each *expression* into the current section. Each *expression* must be a 32-bit value and must evaluate to an integer value. The `.long` directive is not valid for the `.bss` section.

`.popsection`

The `.popsection` directive pops the top of the section stack and continues processing of the popped section.

`.previous`

The `.previous` directive continues processing of the previous section.

`.pushsection section`

The `.pushsection` directive pushes the specified section onto the section stack and switches to another section.

`.quad expression1, expression2, ..., expressionN`

The `.quad` directive generates an initialized word (64-bit, two's complement value) for each *expression* into the current section. Each *expression* must be a 64-bit value, and must evaluate to an integer value. The `.quad` directive is not valid for the `.bss` section.

`.rel symbol@ type`

The `.rel` directive generates the specified relocation entry *type* for the specified *symbol*. The `.lit` directive supports TLS (thread-local storage). Refer to [Chapter 16, “Thread-Local Storage” in Oracle Solaris 11.3 Linkers and Libraries Guide](#) for additional information about TLS.

`.section section, attributes`

The `.section` directive makes *section* the current section. If *section* does not exist, a new section with the specified name and attributes is created. If *section* is a non-reserved section, *attributes* must be included the first time *section* is specified by the `.section` directive.

`.set symbol, expression`

The `.set` directive assigns the value of *expression* to *symbol*. *Expression* can be any legal expression that evaluates to a numerical value.

`.size symbol, expr`

Declares the symbol size to be *expr*. *expr* must be an absolute expression.

`.skip integer, value`

While generating values for any data section, the `.skip` directive causes *integer* bytes to be skipped over, or, optionally, filled with the specified *value*.

`.sleb128 expression`

The `.sleb128` directive generates a signed, little-endian, base 128 number from *expression*.

`.string "string"`

The `.string` directive places the characters in *string* into the object module at the current location and terminates the string with a null byte (`\0`). *String* must be enclosed in double quotes (`"`) (ASCII 0x22). The `.string` directive is not valid for the `.bss` section.

`.symbolic symbol1, symbol2, ..., symbolN`

The `.symbolic` directive declares each *symbol* in the list to have *symbolic* linker scoping. All references to *symbol* within a dynamic module bind to the definition within that module. Outside of the module, *symbol* is treated as global.

`.tbss`

The `.tbss` directive changes the current section to `.tbss`. The `.tbss` section contains uninitialized TLS data objects that will be initialized to zero by the runtime linker.

`.tcomm`

The `.tcomm` directive defines a TLS common block.

**.tdata**

The `.tdata` directive changes the current section to `.tdata`. The `.tdata` section contains the initialization image for initialized TLS data objects.

**.text**

The `.text` directive defines the current section as `.text`.

**.type *symbol* [, *symbol*, ..., *symbol*] , *type* [, *visibility*]**

Declares the type of *symbol*, where *type* can be:

`#object #tls_object #function #no_type`

and where *visibility* can be one of:

`#hidden #protected #eliminate #singleton #exported #internal`

**.uleb128 *expression***

The `.uleb128` directive generates an unsigned, little-endian, base 128 number from *expression*.

**.value *expression1* , *expression2* , ... , *expressionN***

The `.value` directive generates an initialized word (16-bit, two's complement value) for each *expression* into the current section. Each *expression* must be a 16-bit integer value. The `.value` directive is not valid for the `.bss` section.

**.weak *symbol1* , *symbol2* , ... , *symbolN***

The `.weak` directive declares each *symbol* in the argument list to be defined either externally or in the input file and accessible to other files. Default bindings of the symbol are overridden by the `.weak` directive. A *weak* symbol definition in one file satisfies an undefined reference to a global symbol of the same name in another file. Unresolved *weak* symbols have a default value of zero. The link editor does not resolve these symbols. If a *weak* symbol has the same name as a defined *global* symbol, the weak symbol is ignored and no error results. The `.weak` directive does not define the symbol.

**.zero *expression***

While filling a data section, the `.zero` directive fills the number of bytes specified by *expression* with zero (0).





## Instruction Set Mapping

---

This chapter provides a general mapping between the Oracle Solaris x86 assembly language mnemonics and the Intel or Advanced Micro Devices (AMD) mnemonics. Refer to [Table 1, “Instruction References,”](#) on page 34 for details on individual processor instructions.

- “3.1 Instruction Overview” on page 34
- “3.2 General-Purpose Instructions” on page 35
- “3.3 Floating-Point Instructions” on page 47
- “3.4 SIMD State Management Instructions” on page 52
- Table 21, “ADX Instructions,” on page 53
- “3.6 AES Instructions” on page 53
- “3.7 AVX Instructions” on page 54
- “3.8 AVX2 Instructions” on page 77
- “3.9 BMI1 Instructions” on page 85
- “3.10 BMI2 Instructions” on page 85
- “3.11 F16C Instructions” on page 86
- “3.12 FMA Instructions” on page 86
- “3.13 FSGSBASE Instructions” on page 92
- “3.14 MMX Instructions” on page 93
- “3.15 MOVBE Instructions” on page 97
- “3.16 PCLMULQDQ Instructions” on page 97
- “3.17 PREFETCH Instructions” on page 98
- “3.18 RDRAND Instructions” on page 98
- “3.19 RDSEED Instructions” on page 98
- “3.20 SSE Instructions” on page 98
- “3.21 SSE2 Instructions” on page 106
- “3.22 SSE3 Instructions” on page 113
- “3.23 SSE4a Instructions” on page 114
- “3.24 SSE4.1 Instructions” on page 114

- “3.25 SSE4.2 Instructions” on page 117
- “3.26 SSSE3 Instructions” on page 117
- “3.27 Transactional Synchronization Extensions” on page 118
- “3.28 Operating System Support Instructions” on page 119
- “3.29 VMX Instructions” on page 120
- “3.30 XSAVE Instructions” on page 121
- “3.31 3DNow Instructions” on page 121
- “3.32 64-Bit AMD Opteron Considerations” on page 123

## 3.1 Instruction Overview

It is beyond the scope of this manual to document the x86 architecture instruction set. This chapter provides a general mapping between the Oracle Solaris x86 assembly language mnemonics and the Intel or AMD mnemonics to enable you to refer to the Intel or AMD documentation for detailed information about a specific instruction.

Instructions are listed in tables with the following sections:

- Oracle Solaris mnemonic
- Intel/AMD mnemonic
- Description (short)
- Notes
- Reference

The reference column lists the page number and code for the Intel or AMD manual that documents the instruction. See [Table 1, “Instruction References,” on page 34](#) for the codes and links to the associated manuals.

For certain Oracle Solaris mnemonics, the allowed data type suffixes for that mnemonic are indicated in braces ({}), following the mnemonic. For example, `bswap{lq}` indicates that the following mnemonics are valid: `bswap`, `bswapl` (which is the default and equivalent to `bswap`), and `bswapq`. See [“2.2.1 Assembly Language Instructions” on page 22](#) for information on data type suffixes.

**TABLE 1** Instruction References

Manual Code	Name of the Document	Volume	Link
253666-048US/Sep.2013	Intel 64 and IA-32	2A	<a href="#">Instruction Set Reference, A-M</a>

Manual Code	Name of the Document	Volume	Link
	Architectures Software Developer's Manual		
253667-048US/Sep.2013	Intel 64 and IA-32 Architectures Software Developer's Manual	2B	<a href="#">Instruction Set Reference, N-Z</a>
326019-048US/Sep.2013	Intel 64 and IA-32 Architectures Software Developer's Manual	3C	<a href="#">System Programming Guide, Part 3</a>
319433-016/Oct.2013	Intel Architecture Instruction Set Extensions Programming Reference	-	-
AMD: 24594-Rev.3.20-May.2013	AMD64 Architecture Programmer's Manual	3	<a href="#">General-Purpose and System Instructions</a>
AMD: 26568-Rev.3.18-Oct.2013	AMD64 Architecture Programmer's Manual	4	<a href="#">128-Bit and 256-Bit Media Instructions</a>
AMD: 26569-Rev.3.13-May.2013	AMD64 Architecture Programmer's Manual	5	<a href="#">64-Bit Media and x87 Floating-Point Instructions</a>

To locate a specific Oracle Solaris x86 mnemonic, look up the mnemonic in the index.

## 3.2 General-Purpose Instructions

The general-purpose instructions perform basic data movement, memory addressing, arithmetic and logical operations, program flow control, input/output, and string operations on integer, pointer, and BCD data types.

## 3.2.1 Data Transfer Instructions

The data transfer instructions move data between memory and the general-purpose and segment registers, and perform operations such as conditional moves, stack access, and data conversion.

**TABLE 2** Data Transfer Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
bswap{ <i>lq</i> }	BSWAP	byte swap	bswapq valid only under -m64
cbtw	CBW	convert byte to word	
cltd	CDQ	convert doubleword to quadword	%eax → %edx:%eax
cltq	CDQE	convert doubleword to quadword	%eax → %rax cltq valid only under -m64
cmova{ <i>wlq</i> }, cmov { <i>wlq</i> }.a	CMOVA	conditional move if above	cmovaq valid only under -m64
cmovae{ <i>wlq</i> }, cmov { <i>wlq</i> }.ae	CMOVAE	conditional move if above or equal	cmovaeq valid only under -m64
cmovb{ <i>wlq</i> }, cmov { <i>wlq</i> }.b	CMOVB	conditional move if below	cmovbq valid only under -m64
cmovbe{ <i>wlq</i> }, cmov { <i>wlq</i> }.be	CMOVBE	conditional move if below or equal	cmovbeq valid only under -m64
cmovc{ <i>wlq</i> }, cmov { <i>wlq</i> }.c	CMOVC	conditional move if carry	cmovcq valid only under -m64
cmove{ <i>wlq</i> }, cmov { <i>wlq</i> }.e	CMOVE	conditional move if equal	cmoveq valid only under -m64
cmovg{ <i>wlq</i> }, cmov { <i>wlq</i> }.g	CMOVG	conditional move if greater	cmovgq valid only under -m64
cmovge{ <i>wlq</i> }, cmov { <i>wlq</i> }.ge	CMOVGE	conditional move if greater or equal	cmovgeq valid only under -m64
cmovl{ <i>wlq</i> }, cmov { <i>wlq</i> }.l	CMOVL	conditional move if less	cmovlq valid only under -m64
cmovle{ <i>wlq</i> }, cmov { <i>wlq</i> }.le	CMOVLE	conditional move if less or equal	cmovleq valid only under -m64
cmovna{ <i>wlq</i> }, cmov { <i>wlq</i> }.na	CMOVNA	conditional move if not above	cmovnaq valid only under -m64
cmovnae{ <i>wlq</i> }, cmov { <i>wlq</i> }.nae	CMOVNAE	conditional move if not above or equal	cmovnaeq valid only under -m64
cmovnb{ <i>wlq</i> }, cmov { <i>wlq</i> }.nb	CMOVNB	conditional move if not below	cmovnbq valid only under -m64

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>cmovnbe{wlq}, cmov {wlq}.nbe</code>	CMOVNBE	conditional move if not below or equal	<code>cmovnbeq</code> valid only under -m64
<code>cmovnc{wlq}, cmov {wlq}.nc</code>	CMOVNC	conditional move if not carry	<code>cmovncq</code> valid only under -m64
<code>cmovne{wlq}, cmov {wlq}.ne</code>	CMOVNE	conditional move if not equal	<code>cmovneq</code> valid only under -m64
<code>cmovng{wlq}, cmov {wlq}.ng</code>	CMOVNG	conditional move if greater	<code>cmovngq</code> valid only under -m64
<code>cmovnge{wlq}, cmov {wlq}.nge</code>	CMOVNGE	conditional move if not greater or equal	<code>cmovngeq</code> valid only under -m64
<code>cmovnl{wlq}, cmov {wlq}.nl</code>	CMOVNL	conditional move if not less	<code>cmovnlq</code> valid only under -m64
<code>cmovnle{wlq}, cmov {wlq}.nle</code>	CMOVNLE	conditional move if not above or equal	<code>cmovnleq</code> valid only under -m64
<code>cmovno{wlq}, cmov {wlq}.no</code>	CMOVNO	conditional move if not overflow	<code>cmovnoq</code> valid only under -m64
<code>cmovnp{wlq}, cmov {wlq}.np</code>	CMOVNP	conditional move if not parity	<code>cmovnpq</code> valid only under -m64
<code>cmovns{wlq}, cmov {wlq}.ns</code>	CMOVNS	conditional move if not sign (non-negative)	<code>cmovnsq</code> valid only under -m64
<code>cmovnz{wlq}, cmov {wlq}.nz</code>	CMOVNZ	conditional move if not zero	<code>cmovnzq</code> valid only under -m64
<code>cmovo{wlq}, cmov {wlq}.o</code>	CMOVO	conditional move if overflow	<code>cmovoq</code> valid only under -m64
<code>cmovp{wlq}, cmov {wlq}.p</code>	CMOVPE	conditional move if parity	<code>cmovpq</code> valid only under -m64
<code>cmovpe{wlq}, cmov{wlq}. pe</code>	CMOVPE	conditional move if parity even	<code>cmovpeq</code> valid only under -m64
<code>cmovpo{wlq}, cmov{wlq}. po</code>	CMOVPO	conditional move if parity odd	<code>cmovpoq</code> valid only under -m64
<code>cmovs{wlq}, cmov{wlq}.s</code>	CMOVNS	conditional move if sign (negative)	<code>cmovsq</code> valid only under -m64
<code>cmovz{wlq}, cmov{wlq}.z</code>	CMOVZ	conditional move if zero	<code>cmovzq</code> valid only under -m64
<code>cmpxchg{bwlq}</code>	CMPXCHG	compare and exchange	<code>cmpxchgq</code> valid only under -m64
<code>cmpxchg8b</code>	CMPXCHG8B	compare and exchange 8 bytes	
<code>cqtd</code>	CQO	convert quadword to octword	<code>%rax → %rdx:%rax</code>  <code>cqtd</code> valid only under -m64

### 3.2 General-Purpose Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cqto	CQO	convert quadword to octword	%rax → %rdx:%rax cqto valid only under -m64
cwtd	CWD	convert word to doubleword	%ax → %dx:%ax
cwtl	CWDE	convert word to doubleword in %eax register	%ax → %eax
invpcid	INVPCID	Invalidate Process-Context Identifier	page 3-416 (253666-048US/Sep.2013)
mov{bwlq}	MOV	move data between immediate values, general purpose registers, segment registers, and memory	movq valid only under -m64
movabs{bwlq}	MOVABS	move immediate value to register	movabs valid only under -m64
movabs{bwlq}A	MOVABS	move immediate value to register {AL, AX, GAX, RAX}	movabs valid only under -m64
movsb{wlq}, movsw{lq}	MOVSX	move and sign extend	movsbq and movswq valid only under -m64
movzb{wlq}, movzw{lq}	MOVZX	move and zero extend	movzbq and movzwq valid only under -m64
pop{wlq}	POP	pop stack	popq valid only under -m64
popaw	POPA	pop general-purpose registers from stack	popaw invalid under -m64
popal, popa	POPAD	pop general-purpose registers from stack	invalid under -m64
push{wlq}	PUSH	push onto stack	pushq valid only under -m64
pushaw	PUSHA	push general-purpose registers onto stack	pushaw invalid under -m64
pushal, pusha	PUSHAD	push general-purpose registers onto stack	invalid under -m64
xadd{bwlq}	XADD	exchange and add	xaddq valid only under -m64
xchg{bwlq}	XCHG	exchange	xchgq valid only under -m64
xchg{bwlq}A	XCHG	exchange	xchgqA valid only under -m64

## 3.2.2 Binary Arithmetic Instructions

The binary arithmetic instructions perform basic integer computations on operands in memory or the general-purpose registers.

**TABLE 3** Binary Arithmetic Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
adc{bwlq}	ADC	add with carry	adcq valid only under -m64
add{bwlq}	ADD	integer add	addq valid only under -m64
cmp{bwlq}	CMP	compare	cmpq valid only under -m64
dec{bwlq}	DEC	decrement	decq valid only under -m64
div{bwlq}	DIV	divide (unsigned)	divq valid only under -m64
idiv{bwlq}	IDIV	divide (signed)	idivq valid only under -m64
imul{bwlq}	IMUL	multiply (signed)	imulq valid only under -m64
inc{bwlq}	INC	increment	incq valid only under -m64
mul{bwlq}	MUL	multiply (unsigned)	mulq valid only under -m64
neg{bwlq}	NEG	negate	negq valid only under -m64
sbb{bwlq}	SBB	subtract with borrow	sbbq valid only under -m64
sub{bwlq}	SUB	subtract	subq valid only under -m64

## 3.2.3 Decimal Arithmetic Instructions

The decimal arithmetic instructions perform decimal arithmetic on binary coded decimal (BCD) data.

**TABLE 4** Decimal Arithmetic Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
aaa	AAA	ASCII adjust after addition	invalid under -m64
aad	AAD	ASCII adjust before division	invalid under -m64
aam	AAM	ASCII adjust after multiplication	invalid under -m64
aas	AAS	ASCII adjust after subtraction	invalid under -m64

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
daa	DAA	decimal adjust after addition	invalid under -m64
das	DAS	decimal adjust after subtraction	invalid under -m64

## 3.2.4 Logical Instructions

The logical instructions perform basic logical operations on their operands.

**TABLE 5** Logical Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
and{bwlq}	AND	bitwise logical AND	andq valid only under -m64
not{bwlq}	NOT	bitwise logical NOT	notq valid only under -m64
or{bwlq}	OR	bitwise logical OR	orq valid only under -m64
xor{bwlq}	XOR	bitwise logical exclusive OR	xorq valid only under -m64

## 3.2.5 Shift and Rotate Instructions

The shift and rotate instructions shift and rotate the bits in their operands.

**TABLE 6** Shift and Rotate Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
rcl{bwlq}	RCL	rotate through carry left	rclq valid only under -m64
rcr{bwlq}	RCR	rotate through carry right	rcrq valid only under -m64
rol{bwlq}	ROL	rotate left	rolq valid only under -m64
ror{bwlq}	ROR	rotate right	rorq valid only under -m64
sal{bwlq}	SAL	shift arithmetic left	salq valid only under -m64
sar{bwlq}	SAR	shift arithmetic right	sarq valid only under -m64
shl{bwlq}	SHL	shift logical left	shlq valid only under -m64
shld{bwlq}	SHLD	shift left double	shldq valid only under -m64
shr{bwlq}	SHR	shift logical right	shrq valid only under -m64
shrd{bwlq}	SHRD	shift right double	shrdq valid only under -m64



## 3.2.6 Bit and Byte Instructions

The bit instructions test and modify individual bits in operands. The byte instructions set the value of a byte operand to indicate the status of flags in the `%eflags` register.

**TABLE 7** Bit and Byte Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>bsf{wlq}</code>	BSF	bit scan forward	<code>bsfq</code> valid only under <code>-m64</code>
<code>bsr{wlq}</code>	BSR	bit scan reverse	<code>bsrq</code> valid only under <code>-m64</code>
<code>bt{wlq}</code>	BT	bit test	<code>btq</code> valid only under <code>-m64</code>
<code>btc{wlq}</code>	BTC	bit test and complement	<code>btcq</code> valid only under <code>-m64</code>
<code>btr{wlq}</code>	BTR	bit test and reset	<code>btrq</code> valid only under <code>-m64</code>
<code>bts{wlq}</code>	BTS	bit test and set	<code>btsq</code> valid only under <code>-m64</code>
<code>seta</code>	SETA	set byte if above	
<code>setae</code>	SETAE	set byte if above or equal	
<code>setb</code>	SETB	set byte if below	
<code>setbe</code>	SETBE	set byte if below or equal	
<code>setc</code>	SETC	set byte if carry	
<code>sete</code>	SETE	set byte if equal	
<code>setg</code>	SETG	set byte if greater	
<code>setge</code>	SETGE	set byte if greater or equal	
<code>setl</code>	SETL	set byte if less	
<code>setle</code>	SETLE	set byte if less or equal	
<code>setna</code>	SETNA	set byte if not above	
<code>setnae</code>	SETNAE	set byte if not above or equal	
<code>setnb</code>	SETNB	set byte if not below	
<code>setnbe</code>	SETNBE	set byte if not below or equal	
<code>setnc</code>	SETNC	set byte if not carry	
<code>setne</code>	SETNE	set byte if not equal	
<code>setng</code>	SETNG	set byte if not greater	
<code>setnge</code>	SETNGE	set byte if not greater or equal	
<code>setnl</code>	SETNL	set byte if not less	
<code>setnle</code>	SETNLE	set byte if not less or equal	
<code>setno</code>	SETNO	set byte if not overflow	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
setnp	SETNP	set byte if not parity	
setns	SETNS	set byte if not sign (non-negative)	
setnz	SETNZ	set byte if not zero	
seto	SETO	set byte if overflow	
setp	SETP	set byte if parity	
setpe	SETPE	set byte if parity even	
setpo	SETPO	set byte if parity odd	
sets	SETS	set byte if sign (negative)	
setz	SETZ	set byte if zero	
test{bwlq}	TEST	logical compare	testq valid only under -m64

## 3.2.7 Control Transfer Instructions

The control transfer instructions control the flow of program execution.

**TABLE 8** Control Transfer Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
bound{wl}	BOUND	detect value out of range	boundw invalid under -m64
call	CALL	call procedure	
enter	ENTER	high-level procedure entry	
int	INT	software interrupt	
into	INTO	interrupt on overflow	invalid under -m64
iret	IRET	return from interrupt	
ja	JA	jump if above	
jae	JAE	jump if above or equal	
jb	JB	jump if below	
jbe	JBE	jump if below or equal	
jc	JC	jump if carry	
jcxz	JCXZ	jump register %cx zero	
je	JE	jump if equal	
jecxz	JECXZ	jump register %ecx zero	invalid under -m64
jg	JG	jump if greater	
jge	JGE	jump if greater or equal	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
jl	JL	jump if less	
jle	JLE	jump if less or equal	
jmp	JMP	jump	
jnae	JNAE	jump if not above or equal	
jnb	JNB	jump if not below	
jnbe	JNBE	jump if not below or equal	
jnc	JNC	jump if not carry	
jne	JNE	jump if not equal	
jng	JNG	jump if not greater	
jnge	JNGE	jump if not greater or equal	
jnl	JNL	jump if not less	
jnle	JNLE	jump if not less or equal	
jno	JNO	jump if not overflow	
jnp	JNP	jump if not parity	
jns	JNS	jump if not sign (non-negative)	
jnz	JNZ	jump if not zero	
jo	JO	jump if overflow	
jp	JP	jump if parity	
jpe	JPE	jump if parity even	
jpo	JPO	jump if parity odd	
js	JS	jump if sign (negative)	
jz	JZ	jump if zero	
lcall	CALL	call far procedure	valid as indirect only for -m64
leave	LEAVE	high-level procedure exit	
loop	LOOP	loop with %ecx counter	
loope	LOOPE	loop with %ecx and equal	
loopne	LOOPNE	loop with %ecx and not equal	
loopnz	LOOPNZ	loop with %ecx and not zero	
loopz	LOOPZ	loop with %ecx and zero	
lret	RET	return from far procedure	valid as indirect only for m64
ret	RET	return	

## 3.2.8 String Instructions

The string instructions operate on strings of bytes. Operations include storing strings in memory, loading strings from memory, comparing strings, and scanning strings for substrings.

**Note** - The Oracle Solaris mnemonics for certain instructions differ slightly from the Intel/AMD mnemonics. Alphabetization of the table below is by the Oracle Solaris mnemonic. All string operations default to long (doubleword).

**TABLE 9** String Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cmps{q}	CMPS	compare string	cmpsq valid only under -m64
cmpsb	CMPSB	compare byte string	
cmpsl	CMPSD	compare doubleword string	
cmpsw	CMPSW	compare word string	
lods{q}	LODS	load string	lodsq valid only under -m64
lodsb	LODSB	load byte string	
lodsl	LODSD	load doubleword string	
lodsw	LODSW	load word string	
movs{q}	MOVS	move string	movsq valid only under -m64
movsb	MOVSB	move byte string	movsb is not movsb {w q}. See <a href="#">Table 2, “Data Transfer Instructions,” on page 36</a>
movsl, smovl	MOVSD	move doubleword string	
movsw, smovw	MOVSW	move word string	movsw is not movsw {l q}. See <a href="#">Table 2, “Data Transfer Instructions,” on page 36</a>
rep	REP	repeat while %ecx not zero	
repnz	REPNE	repeat while not equal	
repnz	REPZ	repeat while not zero	
repz	REPE	repeat while equal	
repz	REPZ	repeat while zero	
scas{q}	SCAS	scan string	scasq valid only under -m64

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
scasb	SCASB	scan byte string	
scasl	SCASD	scan doubleword string	
scasw	SCASW	scan word string	
stos{q}	STOS	store string	stosq valid only under -m64
stosb	STOSB	store byte string	
stosl	STOSD	store doubleword string	
stosw	STOSW	store word string	

### 3.2.9 I/O Instructions

The input/output instructions transfer data between the processor's I/O ports, registers, and memory.

**TABLE 10** I/O Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
in	IN	read from a port	
ins	INS	input string from a port	
insb	INSB	input byte string from port	
insl	INSD	input doubleword string from port	
insw	INSW	input word string from port	
out	OUT	write to a port	
outs	OUTS	output string to port	
outsb	OUTSB	output byte string to port	
outsl	OUTSD	output doubleword string to port	
outsw	OUTSW	output word string to port	

### 3.2.10 Flag Control (EFLAG) Instructions

The status flag control instructions operate on the bits in the `%eflags` register.

**TABLE 11** Flag Control Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
clc	CLC	clear carry flag	
cld	CLD	clear direction flag	
cli	CLI	clear interrupt flag	
cmc	CMC	complement carry flag	
lahf	LAHF	load flags into %ah register	
popfw	POPF	pop %eflags from stack	
popf{lq}	POPFL	pop %eflags from stack	popfq valid only under -m64
pushfw	PUSHF	push %eflags onto stack	
pushf{lq}	PUSHFL	push %eflags onto stack	pushfq valid only under -m64
sahf	SAHF	store %ah register into flags	
stc	STC	set carry flag	
std	STD	set direction flag	
sti	STI	set interrupt flag	

### 3.2.11 Segment Register Instructions

The segment register instructions load far pointers (segment addresses) into the segment registers.

**TABLE 12** Segment Register Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
lds{wl}	LDS	load far pointer using %ds	lds <sub>l</sub> and lds <sub>w</sub> invalid under -m64
les{wl}	LES	load far pointer using %es	les <sub>l</sub> and les <sub>w</sub> invalid under -m64
lfs{wl}	LFS	load far pointer using %fs	
lgs{wl}	LGS	load far pointer using %gs	
lss{wl}	LSS	load far pointer using %ss	

### 3.2.12 Miscellaneous Instructions

The instructions documented in this section provide a number of useful functions.

**TABLE 13** Miscellaneous Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>cpuid</code>	<code>CPUID</code>	processor identification	
<code>lea{wlq}</code>	<code>LEA</code>	load effective address	<code>leaq</code> valid only under <code>-m64</code>
<code>nop</code>	<code>NOP</code>	no operation	
<code>ud2</code>	<code>UD2</code>	undefined instruction	
<code>xlat</code>	<code>XLAT</code>	table lookup translation	
<code>xlatb</code>	<code>XLATB</code>	table lookup translation	

## 3.3 Floating-Point Instructions

The floating point instructions operate on floating-point, integer, and binary coded decimal (BCD) operands.

### 3.3.1 Data Transfer Instructions (Floating Point)

The data transfer instructions move floating-point, integer, and BCD values between memory and the floating point registers.

**TABLE 14** Data Transfer Instructions (Floating-Point)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>fbld</code>	<code>FBLD</code>	load BCD	
<code>fbstp</code>	<code>FBSTP</code>	store BCD and pop	
<code>fcmovb</code>	<code>FCMOVB</code>	floating-point conditional move if below	
<code>fcmovbe</code>	<code>FCMOVBE</code>	floating-point conditional move if below or equal	
<code>fcmove</code>	<code>FCMOVE</code>	floating-point conditional move if equal	
<code>fcmovnb</code>	<code>FCMOVNB</code>	floating-point conditional move if not below	
<code>fcmovnbe</code>	<code>FCMOVNBE</code>	floating-point conditional move if not below or equal	
<code>fcmovne</code>	<code>FCMOVNE</code>	floating-point conditional move if not equal	
<code>fcmovnu</code>	<code>FCMOVNU</code>	floating-point conditional move if unordered	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fcmovu	FCMOVU	floating-point conditional move if unordered	
fild	FILD	load integer	
fist	FIST	store integer	
fistp	FISTP	store integer and pop	
fld	FLD	load floating-point value	
fst	FST	store floating-point value	
fstp	FSTP	store floating-point value and pop	
fxch	FXCH	exchange registers	

### 3.3.2 Basic Arithmetic Instructions (Floating-Point)

The basic arithmetic instructions perform basic arithmetic operations on floating-point and integer operands.

**TABLE 15** Basic Arithmetic Instructions (Floating-Point)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fabs	FABS	absolute value	
fadd	FADD	add floating-point	
faddp	FADDP	add floating-point and pop	
fchs	FNCHS	change sign	
fdiv	FNDIV	divide floating-point	
fdivp	FNDIVP	divide floating-point and pop	
fdivr	FNDIVR	divide floating-point reverse	
fdivrp	FNDIVRP	divide floating-point reverse and pop	
fiadd	FILDADD	add integer	
fidiv	FILDIV	divide integer	
fidivr	FILDIVR	divide integer reverse	
fimul	FIMUL	multiply integer	
fisub	FISUB	subtract integer	
fisubr	FISUBR	subtract integer reverse	
fmul	FMUL	multiply floating-point	



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fmlp	FMULP	multiply floating-point and pop	
fprem	FPREM	partial remainder	
fprem1	FPREM1	IEEE partial remainder	
frndint	FRNDINT	round to integer	
fscale	FSCALE	scale by power of two	
fsqrt	FSQRT	square root	
fsub	FSUB	subtract floating-point	
fsubp	FSUBP	subtract floating-point and pop	
fsubr	FSUBR	subtract floating-point reverse	
fsubrp	FSUBRP	subtract floating-point reverse and pop	
fextract	FEXTRACT	extract exponent and significand	

### 3.3.3 Comparison Instructions (Floating-Point)

The floating-point comparison instructions operate on floating-point or integer operands.

**TABLE 16** Comparison Instructions (Floating-Point)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fcom	FCOM	compare floating-point	
fcomi	FCOMI	compare floating-point and set %eflags	
fcomip	FCOMIP	compare floating-point, set %eflags, and pop	
fcomp	FCOMP	compare floating-point and pop	
fcompp	FCOMPP	compare floating-point and pop twice	
ficom	FICOM	compare integer	
ficompl	FICOMPL	compare integer and pop	
ftst	FTST	test floating-point (compare with 0.0)	
fucom	FUCOM	unordered compare floating-point	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fucomi	FUCOMI	unordered compare floating-point and set %eflags	
fucomip	FUCOMIP	unordered compare floating-point, set %eflags, and pop	
fucomp	FUCOMP	unordered compare floating-point and pop	
fucompp	FUCOMPP	compare floating-point and pop twice	
fxam	FXAM	examine floating-point	

### 3.3.4 Transcendental Instructions (Floating-Point)

The transcendental instructions perform trigonometric and logarithmic operations on floating-point operands.

**TABLE 17** Transcendental Instructions (Floating-Point)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
f2xm1	F2XM1	computes $2^x-1$	
fcos	FCOS	cosine	
fpatan	FPATAN	partial arctangent	
fptan	FPTAN	partial tangent	
fsin	FSIN	sine	
fsincos	FSINCOS	sine and cosine	
fyl2x	FYL2X	computes $y * \log_2 x$	
fyl2xp1	FYL2XP1	computes $y * \log_2(x+1)$	

### 3.3.5 Load Constants (Floating-Point) Instructions

The load constants instructions load common constants, such as  $\pi$ , into the floating-point registers.

**TABLE 18** Load Constants Instructions (Floating-Point)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fld1	FLD1	load +1.0	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fldl2e	FLDL2E	load $\log_2 e$	
fldl2t	FLDL2T	load $\log_2 10$	
fldlg2	FLDLG2	load $\log_{10} 2$	
fldln2	FLDLN2	load $\log_e 2$	
fldpi	FLDPI	load $\pi$	
fldz	FLDZ	load +0.0	

### 3.3.6 Control Instructions (Floating-Point)

The floating-point control instructions operate on the floating-point register stack and save and restore the floating-point state.

**TABLE 19** Control Instructions (Floating-Point)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fclex	FCLEX	clear floating-point exception flags after checking for error conditions	
fdecstp	FDECSTP	decrement floating-point register stack pointer	
ffree	FFREE	free floating-point register	
fincstp	FINCSTP	increment floating-point register stack pointer	
finit	FINIT	initialize floating-point unit after checking error conditions	
fldcw	FLDCW	load floating-point unit control word	
fldenv	FLDENV	load floating-point unit environment	
fnclex	FNCLEX	clear floating-point exception flags without checking for error conditions	
fninit	FNINIT	initialize floating-point unit without checking error conditions	
fnop	FNOP	floating-point no operation	

### 3.4 SIMD State Management Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fnsave	FNSAVE	save floating-point unit state without checking error conditions	
fnstcw	FNSTCW	store floating-point unit control word without checking error conditions	
fnstenv	FNSTENV	store floating-point unit environment without checking error conditions	
fnstsw	FNSTSW	store floating-point unit status word without checking error conditions	
frstor	FRSTOR	restore floating-point unit state	
fsave	FSAVE	save floating-point unit state after checking error conditions	
fstcw	FSTCW	store floating-point unit control word after checking error conditions	
fstenv	FSTENV	store floating-point unit environment after checking error conditions	
fstsw	FSTSW	store floating-point unit status word after checking error conditions	
fwait	FWAIT	wait for floating-point unit	
wait	WAIT	wait for floating-point unit	

## 3.4 SIMD State Management Instructions

The `fxsave` and `fxrstor` instructions save and restore the state of the floating-point unit and the MMX, XMM, and MXCSR registers.

**TABLE 20** SIMD State Management Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
fxrstor	FXRSTOR	restore floating-point unit and SIMD state	
fxsave	FXSAVE	save floating-point unit and SIMD state	

## 3.5 ADX Instructions

**TABLE 21** ADX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
adcx	ADCX	Unsigned Integer Addition of Two Operands With Carry Flag	page 93-94 (325383-053US/Jan.2015)
adox	ADOX	Unsigned Integer Addition of Two Operands With Overflow Flag	page 108-109 (325383-053US/Jan.2015)

## 3.6 AES Instructions

**TABLE 22** AES Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
aesdec	AESDEC	Perform One Round of an AES Decryption Flow	page 3-40 (253666-048US/Sep.2013)
aesdeclast	AESDECLAST	Perform Last Round of an AES Decryption Flow	page 3-42 (253666-048US/Sep.2013)
aesenc	AESENC	Perform One Round of an AES Encryption Flow	page 3-44 (253666-048US/Sep.2013)
aesenclast	AESENCLAST	Perform Last Round of an AES Encryption Flow	page 3-46 (253666-048US/Sep.2013)
aesimc	AESIMC	Perform the AES InvMixColumn Transformation	page 3-48 (253666-048US/Sep.2013)
aeskeygenassist	AESKEYGENASSIST	AES Round Key Generation Assist	page 3-49 (253666-048US/Sep.2013)

### 3.6.1 Advanced Vector Extensions of AES Instructions

**TABLE 23** Advanced Vector Extensions of AES Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vaesdec	AESDEC	Perform One Round of an AES Decryption Flow	page 3-40 (253666-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vaesdeclast	AESDECLAST	Perform Last Round of an AES Decryption Flow	page 3-42 (253666-048US/Sep.2013)
vaesenc	AESENC	Perform One Round of an AES Encryption Flow	page 3-44 (253666-048US/Sep.2013)
vaesenclast	AESENCLAST	Perform Last Round of an AES Encryption Flow	page 3-46 (253666-048US/Sep.2013)
vaesimc	AESIMC	Perform the AES InvMixColumn Transformation	page 3-48 (253666-048US/Sep.2013)
vaeskeygenassist	AESKEYGENASSIST	AES Round Key Generation Assist	page 3-49 (253666-048US/Sep.2013)

## 3.7 AVX Instructions

**TABLE 24** AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vaddpd	ADDPD	Add Packed Double-Precision Floating-Point Values	page 5-7 (319433-016/Oct.2013)
vaddps	ADDPS	Add Packed Single-Precision Floating-Point Values	page 5-10 (319433-016/Oct.2013)
vaddsd	ADDSD	Add Scalar Double-Precision Floating-Point Values	page 5-13 (319433-016/Oct.2013)
vaddss	ADDSS	Add Scalar Single-Precision Floating-Point Values	page 5-15 (319433-016/Oct.2013)
vaddsubpd	ADDSUBPD	Packed Double-FP Add/Subtract	page 3-35 (253666-048US/Sep.2013)
vaddsubps	ADDSUBPS	Packed Single-FP Add/Subtract	page 3-37 (253666-048US/Sep.2013)
vandnpd	ANDNPD	Bitwise Logical AND NOT of Packed Double-Precision Floating-Point Values	page 3-58 (253666-048US/Sep.2013)
vandnps	ANDNPS	Bitwise Logical AND NOT of Packed Single-Precision Floating-Point Values	page 3-60 (253666-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vandpd	ANDPD	Bitwise Logical AND of Packed Double-Precision Floating-Point Values	page 3-54 (253666-048US/Sep.2013)
vandps	ANDPS	Bitwise Logical AND of Packed Single-Precision Floating-Point Values	page 3-56 (253666-048US/Sep.2013)
vblendpd	BLENDPD	Blend Packed Double Precision Floating-Point Values	page 3-64 (253666-048US/Sep.2013)
vblendps	BLENDPS	Blend Packed Single Precision Floating-Point Values	page 3-68 (253666-048US/Sep.2013)
vblendvpd	BLENDVPD	Variable Blend Packed Double Precision Floating-Point Values	page 3-70 (253666-048US/Sep.2013)
vblendvps	BLENDVPS	Variable Blend Packed Single Precision Floating-Point Values	page 3-72 (253666-048US/Sep.2013)
vcmpcq_ospd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpcq_uqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpcq_uspd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpcqpdp	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpfalse_ospd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpfalsepdp	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpge_oqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpgepdp	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpgt_oqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpgtpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmple_oqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmplepd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmlt_oqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmltpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpneq_oqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpneq_ospd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpneq_uspd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpneqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpnge_uqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpngepd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpngt_uqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpngtpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpnle_uqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpnlepd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpnlt_uqpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpnltpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpord_spd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpordpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmppd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmptrue_uspd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmptruepd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpunord_spd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpunordpd	CMPPD	Compare Packed Double-Precision Floating-Point Values	page 5-40 (319433-016/Oct. 2013)
vcmpcq_osp	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpcq_uqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpcq_usps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpcqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpfalse_osp	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpfalseps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpge_oqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpgeps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpgt_oqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpgtps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmple_oqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpleps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmlt_oqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmltps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpneq_oqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpneq_osps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpneq_usps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpneqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpnge_uqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpngeps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpngt_uqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpngtps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpnle_uqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpnleps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpnlt_uqps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpnltps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpord_sps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpordps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmppps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmptrue_usps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmptrueps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpunord_sps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmpunordps	CMPPS	Compare Packed Single-Precision Floating-Point Values	page 5-46 (319433-016/Oct. 2013)
vcmppeq_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmppeq_uqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmppeq_ussd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpfalse_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpfalsesd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpge_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpgesd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpgt_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpgtsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmple_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmplesd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmplt_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpltsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpneq_ossd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpneq_ussd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpneqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpnge_uqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpngesd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpngt_uqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpngtsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpnle_uqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpnlesd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpnlt_uqsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpnltsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpord_ssd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpordsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmptrue_ussd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmptruesd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpunord_ssd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)
vcmpunordsd	CMPSD	Compare Scalar Double-Precision Floating-Point Value	page 5-52 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpcq_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpcq_uqss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpcq_uss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpcqs	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpfalse_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpfalse	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpge_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpge	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpgt_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpgt	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpgtss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmple_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpless	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmlt_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmltss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpneq_oss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmpneq_oss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpneq_uss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpneqss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpnge_uqss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpngess	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpngt_uqss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpngtss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpnle_uqss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpnless	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpnlt_uqss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpnltss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpord_oss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpordss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmptrue_uss	CMPSS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcmptruess	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpunord_sss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcmpunordss	CMPS	Compare Scalar Single-Precision Floating-Point Value	page 5-57 (319433-016/Oct. 2013)
vcomisd	COMISD	Compare Scalar Ordered Double-Precision Floating-Point Values and Set EFLAGS	page 5-62 (319433-016/Oct. 2013)
vcomiss	COMISS	Compare Scalar Ordered Single-Precision Floating-Point Values and Set EFLAGS	page 5-64 (319433-016/Oct. 2013)
vcvtdq2pd	CVTDQ2PD	Convert Packed Doubleword Integers to Packed Double-Precision Floating-Point Values	page 5-79 (319433-016/Oct. 2013)
vcvtdq2ps	CVTDQ2PS	Convert Packed Doubleword Integers to Packed Single-Precision Floating-Point Values	page 5-82 (319433-016/Oct. 2013)
vcvtpd2dq( x y)	CVTPD2DQ	Convert Packed Double-Precision Floating-Point Values to Packed Doubleword Integers	page 5-85 (319433-016/Oct. 2013)
vcvtpd2ps( x y)	CVTPD2PS	Convert Packed Double-Precision Floating-Point Values to Packed Single-Precision Floating-Point Values 5-88(319433-016/Oct.2013)	page 5-88 (319433-016/Oct. 2013)
vcvtps2dq	CVTPS2DQ	Convert Packed Single-Precision Floating-Point Values to Packed Signed Doubleword Integer Values	page 5-100 (319433-016/Oct. 2013)
vcvtps2pd	CVTPS2PD	Convert Packed Single-Precision Floating-Point Values to Packed Double-Precision Floating-Point	page (319433-016/Oct.2013)
vcvtsd2si( q l)	CVTSD2SI	Convert Scalar Double-Precision Floating-Point Value to Doubleword Integer	page 5-108 (319433-016/Oct. 2013)



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
<code>vcvtisd2ss</code>	<code>CVTSD2SS</code>	Convert Scalar Double-Precision Floating-Point Value to Scalar Single-Precision Floating-Point Value	page 5-112 (319433-016/Oct. 2013)
<code>vcvtisi2sd( q l)</code>	<code>CVTISI2SD</code>	Convert Doubleword Integer to Scalar Double-Precision Floating-Point Value	page 5-114 (319433-016/Oct. 2013)
<code>vcvtisi2ss( q l)</code>	<code>CVTISI2SS</code>	Convert Doubleword Integer to Scalar Single-Precision Floating-Point Value	page 5-116 (319433-016/Oct. 2013)
<code>vcvtss2sd</code>	<code>CVTSS2SD</code>	Convert Scalar Single-Precision Floating-Point Value to Scalar Double-Precision Floating-Point Value	page 5-118 (319433-016/Oct. 2013)
<code>vcvtss2si( q l)</code>	<code>CVTSS2SI</code>	Convert Scalar Single-Precision Floating-Point Value to Doubleword Integer	page 5-120 (319433-016/Oct. 2013)
<code>vcvttpd2dq( x y)</code>	<code>CVTTPD2DQ</code>	Convert with Truncation Packed Double-Precision Floating-Point Values to Packed Doubleword	page (319433-016/Oct.2013)
<code>vcvttps2dq</code>	<code>CVTTPS2DQ</code>	Convert with Truncation Packed Single-Precision Floating-Point Values to Packed Signed Doubleword	page (319433-016/Oct.2013)
<code>vcvttsd2si( q l)</code>	<code>CVTTS2SI</code>	Convert with Truncation Scalar Double-Precision Floating-Point Value to Signed Integer	page 5-134 (319433-016/Oct. 2013)
<code>vcvtts2si( q l)</code>	<code>CVTTS2SI</code>	Convert with Truncation Scalar Single-Precision Floating-Point Value to Integer	page 5-137 (319433-016/Oct. 2013)
<code>vdivpd</code>	<code>DIVPD</code>	Divide Packed Double-Precision Floating-Point Values	page 5-66 (319433-016/Oct. 2013)
<code>vdivps</code>	<code>DIVPS</code>	Divide Packed Single-Precision Floating-Point Values	page 5-68 (319433-016/Oct. 2013)
<code>vdivsd</code>	<code>DIVSD</code>	Divide Scalar Double-Precision Floating-Point Value	page 5-71 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vdivss	DIVSS	Divide Scalar Single-Precision Floating-Point Values	page 5-73 (319433-016/Oct. 2013)
vdppd	DPPD	Dot Product of Packed Double Precision Floating-Point Values	page 3-240 (253666-048US/Sep.2013)
vdpps	DPPS	Dot Product of Packed Single Precision Floating-Point Values	page 3-242 (253666-048US/Sep.2013)
vextractps	EXTRACTPS	Extract Packed Floating-Point Values	page 5-158 (319433-016/Oct. 2013)
vhaddpd	HADDPD	Packed Double-FP Horizontal Add	page 3-370 (253666-048US/Sep.2013)
vhaddps	HADDP S	Packed Single-FP Horizontal Add	page 3-373 (253666-048US/Sep.2013)
vhsubpd	HSUBPD	Packed Double-FP Horizontal Subtract	page 3-377 (253666-048US/Sep.2013)
vhsubps	HSUBPS	Packed Single-FP Horizontal Subtract	page 3-380 (253666-048US/Sep.2013)
vinsertps	INSERTPS	Insert Scalar Single-Precision Floating-Point Value	page 5-311 (319433-016/Oct. 2013)
vlddqu	LDDQU	Load Unaligned Integer 128 Bits	page 3-444 (253666-048US/Sep.2013)
vldmxcsr	LDMXCSR	Load MXCSR Register	page 3-446 (253666-048US/Sep.2013)
vmaskmovdqu	MASKMOVDQU	Store Selected Bytes of Double Quadword	page 3-478 (253666-048US/Sep.2013)
vmaxpd	MAXPD	Maximum of Packed Double-Precision Floating-Point Values	page 5-314 (319433-016/Oct. 2013)
vmaxps	MAXPS	Maximum of Packed Single-Precision Floating-Point Values	page 5-317 (319433-016/Oct. 2013)
vmaxsd	MAXSD	Return Maximum Scalar Double-Precision Floating-Point Value	page 5-320 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vmaxss	MAXSS	Return Maximum Scalar Single-Precision Floating-Point Value	page 5-322 (319433-016/Oct. 2013)
vminpd	MINPD	Minimum of Packed Double-Precision Floating-Point Values	page 5-324 (319433-016/Oct. 2013)
vminps	MINPS	Minimum of Packed Single-Precision Floating-Point Values	page 5-327 (319433-016/Oct. 2013)
vminsd	MINSF	Return Minimum Scalar Double-Precision Floating-Point Value	page 5-330 (319433-016/Oct. 2013)
vminss	MINSS	Return Minimum Scalar Single-Precision Floating-Point Value	page 5-332 (319433-016/Oct. 2013)
vmovapd	MOVAPD	Move Aligned Packed Double-Precision Floating-Point Values	page 5-334 (319433-016/Oct. 2013)
vmovaps	MOVAPS	Move Aligned Packed Single-Precision Floating-Point Values	page 5-337 (319433-016/Oct. 2013)
vmov(q d)	MOVDMOVQ	Move Doubleword and Quadword	page 5-340 (319433-016/Oct. 2013)
vmovddup	MOVDDUP	Replicate Double FP Values	page 5-346 (319433-016/Oct. 2013)
vmovdqa	MOVDQA	Move Aligned Packed Integer Values	page 5-349 (319433-016/Oct. 2013)
vmovdqu	MOVDQU VMOVDQU32 VMOVDQU64	Move Unaligned Packed Integer Values	page 5-353 (319433-016/Oct. 2013)
vmovhlps	MOVHPLS	Move Packed Single-Precision Floating-Point Values High to Low	page 5-357 (319433-016/Oct. 2013)
vmovhpd	MOVHPD	Move High Packed Double-Precision Floating-Point Values	page 5-359 (319433-016/Oct. 2013)
vmovhps	MOVHPS	Move High Packed Single-Precision Floating-Point Values	page 5-361 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vmovlhps	MOVLHPS	Move Packed Single-Precision Floating-Point Values Low to High	page 5-363 (319433-016/Oct. 2013)
vmovlpd	MOVLPD	Move Low Packed Double-Precision Floating-Point Values	page 5-365 (319433-016/Oct. 2013)
vmovlps	MOVLPS	Move Low Packed Single-Precision Floating-Point Values	page 5-367 (319433-016/Oct. 2013)
vmovmskpd	MOVMSKPD	Extract Packed Double-Precision Floating-Point Sign Mask	page 3-539 (253666-048US/Sep.2013)
vmovmskps	MOVMSKPS	Extract Packed Single-Precision Floating-Point Sign Mask	page 3-541 (253666-048US/Sep.2013)
vmovntdq	MOVNTDQ	Store Packed Integers Using Non-Temporal Hint	page 5-371 (319433-016/Oct. 2013)
vmovntdqa	MOVNTDQA	Load Double Quadword Non-Temporal Aligned Hint	page 5-369 (319433-016/Oct. 2013)
vmovntpd	MOVNTPD	Store Packed Double-Precision Floating-Point Values Using Non-Temporal Hint	page 5-373 (319433-016/Oct. 2013)
vmovntps	MOVNTPS	Store Packed Single-Precision Floating-Point Values Using Non-Temporal Hint	page 5-375 (319433-016/Oct. 2013)
vmovq	MOVQ	Move Quadword	page 5-343 (319433-016/Oct. 2013)
vmovsd	MOVSD	Move or Merge Scalar Double-Precision Floating-Point Value	page 5-377 (319433-016/Oct. 2013)
vmovshdup	MOVSHDUP	Replicate Single FP Values	page 5-380 (319433-016/Oct. 2013)
vmovsldup	MOVSLDUP	Replicate Single FP Values	page 5-383 (319433-016/Oct. 2013)
vmovss	MOVSS	Move or Merge Scalar Single-Precision Floating-Point Value	page 5-386 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vmovupd	MOVUPD	Move Unaligned Packed Double-Precision Floating-Point Values	page 5-389 (319433-016/Oct. 2013)
vmovups	MOVUPS	Move Unaligned Packed Single-Precision Floating-Point Values	page 5-392 (319433-016/Oct. 2013)
vmpsadbw	MPSADBW	Compute Multiple Packed Sums of Absolute Difference	page 3-577 (253666-048US/Sep.2013)
vmulpd	MULPD	Multiply Packed Double-Precision Floating-Point Values	page 5-395 (319433-016/Oct. 2013)
vmulps	MULPS	Multiply Packed Single-Precision Floating-Point Values	page 5-397 (319433-016/Oct. 2013)
vmulsd	MULSD	Multiply Scalar Double-Precision Floating-Point Value	page 5-400 (319433-016/Oct. 2013)
vmulss	MULSS	Multiply Scalar Single-Precision Floating-Point Values	page 5-402 (319433-016/Oct. 2013)
vorpd	ORPD	Bitwise Logical OR of Double-Precision Floating-Point Values	page 4-13 (253667-048US/Sep.2013)
vorps	ORPS	Bitwise Logical OR of Single-Precision Floating-Point Values	page 4-15 (253667-048US/Sep.2013)
vpabs(w b d)	PABSB PABSW PABSD PABSQ	Packed Absolute Value	page 5-404 (319433-016/Oct. 2013)
vpackss(dw wb)	PACKSSWB PACKSSDW	Pack with Signed Saturation	page 4-27 (253667-048US/Sep.2013)
vpackusdw	PACKUSDW	Pack with Unsigned Saturation	page 4-32 (253667-048US/Sep.2013)
vpackuswb	PACKUSWB	Pack with Unsigned Saturation	page 4-35 (253667-048US/Sep.2013)
vpadd(q w b d)	PADDB PADDW	Add Packed Integers	page 5-408 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	PADD PADDQ		
vpadds(w b)	PADD SB PADD SW	Add Packed Signed Integers with Signed Saturation	page 4-44 (253667-048US/ Sep.2013)
vpaddus(w b)	PADD USB PADD USW	Add Packed Unsigned Integers with Unsigned Saturation	page 4-47 (253667-048US/ Sep.2013)
vpalignr	PALIGNR	Packed Align Right	page 4-50 (253667-048US/ Sep.2013)
vpand	PAND	Logical AND	page 5-413 (319433-016/Oct. 2013)
vpandn	PANDN	Logical AND NOT	page 5-416 (319433-016/Oct. 2013)
vpavg(w b)	PAVGB PAVGW	Average Packed Integers	page 4-58 (253667-048US/ Sep.2013)
vpblendvb	PBLENDVB	Variable Blend Packed Bytes	page 4-61 (253667-048US/ Sep.2013)
vpblendw	PBLENDW	Blend Packed Words	page 4-65 (253667-048US/ Sep.2013)
vpcmpcq(q w b d)	PCMPQ B PCMPQ W PCMPQ D PCMPQ Q	Compare Packed Integers for Equality	page 5-419 (319433-016/Oct. 2013)
vpcmpetri	PCMPETRI	Packed Compare Explicit Length Strings, Return Index	page 4-77 (253667-048US/ Sep.2013)
vpcmpetrm	PCMPETRM	Packed Compare Explicit Length Strings, Return Mask	page 4-79 (253667-048US/ Sep.2013)
vpcmpgt(q w b d)	PCMPGT B PCMPGT W PCMPGT D PCMPGT Q	Compare Packed Integers for Greater Than	page 5-424 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vpcmpistri	PCMPISTRI	Packed Compare Implicit Length Strings, Return Index	page 4-87 (253667-048US/ Sep.2013)
vpcmpistrm	PCMPISTRM	Packed Compare Implicit Length Strings, Return Mask	page 4-89 (253667-048US/ Sep.2013)
vpextr(q b d)	PEXTRB PEXTRD PEXTRQ	Extract Byte/Dword/Qword	page 4-95 (253667-048US/ Sep.2013)
vpextrw	PEXTRW	Extract Word	page 4-98 (253667-048US/ Sep.2013)
vphaddsw	PHADDSW	Packed Horizontal Add and Saturate	page 4-105 (253667-048US/ Sep.2013)
vphadd(w d)	PHADDW PHADD	Packed Horizontal Add	page 4-101 (253667-048US/ Sep.2013)
vphminposuw	PHMINPOSUW	Packed Horizontal Word Minimum	page 4-107 (253667-048US/ Sep.2013)
vphsubsw	PHSUBSW	Packed Horizontal Subtract and Saturate	page 4-112 (253667-048US/ Sep.2013)
vphsub(w d)	PHSUBW PHSUBD	Packed Horizontal Subtract	page 4-109 (253667-048US/ Sep.2013)
vpinsr(q b w d)	PINSRB PINSRD PINSRQ	Insert Byte/Dword/Qword	page 4-114 (253667-048US/ Sep.2013)
vpinsrw	PINSRW	Insert Word	page 4-116 (253667-048US/ Sep.2013)
vpmaddubsw	PMADDUBSW	Multiply and Add Packed Signed and Unsigned Bytes	page 4-118 (253667-048US/ Sep.2013)
vpmaddwd	PMADDWD	Multiply and Add Packed Integers	page 4-120 (253667-048US/ Sep.2013)
vpmaxs(w b d)	PMAXSB PMAXSW	Maximum of Packed Signed Integers	page 5-471 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	PMAXSD PMAXSQ		
vpmaxub	PMAXUB	Maximum of Packed Unsigned Byte Integers	page 4-131 (253667-048US/ Sep.2013)
vpmaxud	PMAXUD PMAXUQ	Maximum of Packed Unsigned Integers	page 5-476 (319433-016/Oct. 2013)
vpmaxuw	PMAXUW	Maximum of Packed Word Integers	page 4-136 (253667-048US/ Sep.2013)
vpminsb	PMINSB	Minimum of Packed Signed Byte Integers	page 4-138 (253667-048US/ Sep.2013)
vpminsd	PMINSD PMINSQ	Minimum of Packed Signed Integers	page 5-479 (319433-016/Oct. 2013)
vpminsw	PMINSW	Minimum of Packed Signed Word Integers	page 4-143 (253667-048US/ Sep.2013)
vpminub	PMINUB	Minimum of Packed Unsigned Byte Integers	page 4-146 (253667-048US/ Sep.2013)
vpminud	PMINUD PMINUQ	Minimum of Packed Unsigned Integers	page 5-482 (319433-016/Oct. 2013)
vpminuw	PMINUW	Minimum of Packed Word Integers	page 4-151 (253667-048US/ Sep.2013)
vpmovmskb	PMOVMSKB	Move Byte Mask	page 4-153 (253667-048US/ Sep.2013)
vpmovsx(bd bq bw dq wd wq)	PMOVSX	Packed Move with Sign Extend	page 5-500 (319433-016/Oct. 2013)
vpmovzx(bd bq bw dq wd wq)	PMOVZX	Packed Move with Zero Extend	page 5-507 (319433-016/Oct. 2013)
vpmuldq	PMULDQ	Multiply Packed Doubleword Integers	page 5-514 (319433-016/Oct. 2013)
vpmulhrsw	PMULHRSW	Packed Multiply High with Round and Scale	page 4-165 (253667-048US/ Sep.2013)



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vpmulhuw	PMULHUW	Multiply Packed Unsigned Integers and Store High Result	page 4-168 (253667-048US/ Sep.2013)
vpmulhw	PMULHW	Multiply Packed Signed Integers and Store High Result	page 4-172 (253667-048US/ Sep.2013)
vpmulld	PMULLD	Multiply Packed Integers and Store Low Result	page 5-516 (319433-016/Oct. 2013)
vpmullw	PMULLW	Multiply Packed Signed Integers and Store Low Result	page 4-177 (253667-048US/ Sep.2013)
vpmuludq	PMULUDQ	Multiply Packed Unsigned Doubleword Integers	page 5-519 (319433-016/Oct. 2013)
vpor	POR	Bitwise Logical Or	page 5-521 (319433-016/Oct. 2013)
vpsadbw	PSADBW	Compute Sum of Absolute Differences	page 4-198 (253667-048US/ Sep.2013)
vpshufb	PSHUFB	Packed Shuffle Bytes	page 4-201 (253667-048US/ Sep.2013)
vpshufd	PSHUFD	Shuffle Packed Doublewords	page 5-533 (319433-016/Oct. 2013)
vpshufhw	PSHUFHW	Shuffle Packed High Words	page 4-206 (253667-048US/ Sep.2013)
vpshuflw	PSHUFLW	Shuffle Packed Low Words	page 4-208 (253667-048US/ Sep.2013)
vpsign(w b d)	PSIGNB PSIGNW PSIGND	Packed SIGN	page 4-211 (253667-048US/ Sep.2013)
vpslldq	PSLLDQ	Shift Double Quadword Left Logical	page 4-215 (253667-048US/ Sep.2013)
vpsll(q w d)	PSLLW PSLLD PSLLQ	Bit Shift Left	page 5-536 (319433-016/Oct. 2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vpsra(w d)	PSRAW	Bit Shift Arithmetic Right	page 5-544 (319433-016/Oct. 2013)
vpsrldq	PSRLDQ	Shift Double Quadword Right Logical	page 4-228 (253667-048US/Sep.2013)
vpsrl(q w d)	PSRLW PSRLD PSRLQ	Shift Packed Data Right Logical	page 5-550 (319433-016/Oct. 2013)
vpsub(q w b d)	PSUBB PSUBW PSUBD PSUBQ	Packed Integer Subtract	page 5-563 (319433-016/Oct. 2013)
vpsubs(w b)	PSUBSB PSUBSW	Subtract Packed Signed Integers with Signed Saturation	page 4-243 (253667-048US/Sep.2013)
vpsubus(w b)	PSUBUSB PSUBUSW	Subtract Packed Unsigned Integers with Unsigned Saturation	page 4-246 (253667-048US/Sep.2013)
vptest	PTEST	Logical Compare	page 4-249 (253667-048US/Sep.2013)
vpunpckh(bw dq qdq wd)	PUNPCKHBW PUNPCKHWD PUNPCKHDQ PUNPCKHQDQ	Unpack High Data	page 5-571 (319433-016/Oct. 2013)
vpunpckl(bw dq qdq wd)	PUNPCKLBW PUNPCKLWD PUNPCKLDQ PUNPCKLQDQ	Unpack Low Data	page 5-578 (319433-016/Oct. 2013)
vpxor	PXOR PXORD PXORQ	Exclusive Or	page 5-612 (319433-016/Oct. 2013)
vrcpps	RCPPS	Compute Reciprocals of Packed Single-Precision Floating-Point Values	page 4-280 (253667-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vrcpss	RCPSS	Compute Reciprocal of Scalar Single-Precision Floating-Point Values	page 4-282 (253667-048US/ Sep.2013)
vroundpd	ROUNDPD	Round Packed Double Precision Floating-Point Values	page 4-312 (253667-048US/ Sep.2013)
vroundps	ROUNDPS	Round Packed Single Precision Floating-Point Values	page 4-315 (253667-048US/ Sep.2013)
vroundsd	ROUNDSD	Round Scalar Double Precision Floating-Point Values	page 4-318 (253667-048US/ Sep.2013)
vroundss	ROUNDSS	Round Scalar Single Precision Floating-Point Values	page 4-320 (253667-048US/ Sep.2013)
vrsqrtps	RSQRTPS	Compute Reciprocals of Square Roots of Packed Single-Precision Floating-Point Values	page 4-324 (253667-048US/ Sep.2013)
vrsqrtss	RSQRTSS	Compute Reciprocal of Square Root of Scalar Single-Precision Floating-Point Value	page 4-326 (253667-048US/ Sep.2013)
vshufpd	SHUFPD	Shuffle Packed Double-Precision Floating-Point Values	page 5-589 (319433-016/Oct. 2013)
vshufps	SHUFPS	Shuffle Packed Single-Precision Floating-Point Values	page 5-593 (319433-016/Oct. 2013)
vsqrtpd	SQRTPD	Square Root of Double-Precision Floating-Point Values	page 5-597 (319433-016/Oct. 2013)
vsqrtps	SQRTPS	Square Root of Single-Precision Floating-Point Values	page 5-599 (319433-016/Oct. 2013)
vsqrtsd	SQRSD	Compute Square Root of Scalar Double-Precision Floating-Point Value	page 5-601 (319433-016/Oct. 2013)
vsqrtss	SQRSS	Compute Square Root of Scalar Single-Precision Value	page 5-603 (319433-016/Oct. 2013)
vstmcsr	STMCSR	Store MXCSR Register State	page 4-378 (253667-048US/ Sep.2013)

### 3.7 AVX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vsubpd	SUBPD	Subtract Packed Double-Precision Floating-Point Values	page 5-656 (319433-016/Oct. 2013)
vsubps	SUBPS	Subtract Packed Single-Precision Floating-Point Values	page 5-659 (319433-016/Oct. 2013)
vsubsd	SUBSD	Subtract Scalar Double-Precision Floating-Point Value	page 5-662 (319433-016/Oct. 2013)
vsubss	SUBSS	Subtract Scalar Single-Precision Floating-Point Value	page 5-664 (319433-016/Oct. 2013)
vucomisd	UCOMISD	Unordered Compare Scalar Double-Precision Floating-Point Values and Set EFLAGS	page 5-666 (319433-016/Oct. 2013)
vucomiss	UCOMISS	Unordered Compare Scalar Single-Precision Floating-Point Values and Set EFLAGS	page 5-668 (319433-016/Oct. 2013)
vunpckhpd	UNPCKHPD	Unpack and Interleave High Packed Double-Precision Floating-Point Values	page 5-670 (319433-016/Oct. 2013)
vunpckhps	UNPCKHPS	Unpack and Interleave High Packed Single-Precision Floating-Point Values	page 5-673 (319433-016/Oct. 2013)
vunpcklpd	UNPCKLPD	Unpack and Interleave Low Packed Double-Precision Floating-Point Values	page 5-677 (319433-016/Oct. 2013)
vunpcklps	UNPCKLPS	Unpack and Interleave Low Packed Single-Precision Floating-Point Values	page 5-680 (319433-016/Oct. 2013)
vbroadcast(f128 sd ss)	VBROADCAST	Load with Broadcast Floating-Point Data	page 5-27 (319433-016/Oct. 2013)
vextractf128	VEEXTRACTF128 VEEXTRACTF32x4 VEEXTRACTF64x4	Extract Packed Floating-Point Values	page 5-152 (319433-016/Oct. 2013)
vinserf128	VINSERTF128 VINSERTF32x4 VINSERTF64x4	Insert Packed Floating-Point Values	page 5-305 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vmaskmov (pd ps)	VMAKMOV	Conditional SIMD Packed Loads and Stores	page 4-506 (253667-048US/Sep.2013)
vperm2f128	VPERM2F128	Permute Floating-Point Values	page 4-527 (253667-048US/Sep.2013)
vpermilpd	VPERMILPD	Permute Double-Precision Floating-Point Values	page 5-445 (319433-016/Oct. 2013)
vpermilps	VPERMILPS	Permute Single-Precision Floating-Point Values	page 5-450 (319433-016/Oct. 2013)
vtestp(d s)	VTESTPDVTESTPS	Packed Bit Test	page 4-538 (253667-048US/Sep.2013)
vzeroall	VZEROALL	Zero All YMM Registers	page 4-541 (253667-048US/Sep.2013)
vzeroupper	VZERoupper	Zero Upper Bits of YMM Registers	page 4-543 (253667-048US/Sep.2013)
vxorpd	XORPD	Bitwise Logical XOR for Double-Precision Floating-Point Values	page 4-572 (253667-048US/Sep.2013)
vxorps	XORPS	Bitwise Logical XOR for Single-Precision Floating-Point Values	page 4-574 (253667-048US/Sep.2013)
vpclmulqdq	PCLMULQDQ	Carry-Less Multiplication Quadword Requires PCLMULQDQ CPUID-flag	page 4-68 (253667-048US/Sep.2013)

## 3.8 AVX2 Instructions

**TABLE 25** AVX2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vmovntdqa	MOVNTDQA	Load Double Quadword Non-Temporal Aligned Hint	page 5-369 (319433-016/Oct. 2013)

### 3.8 AVX2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vmpsadbw	MPSADBW	Compute Multiple Packed Sums of Absolute Difference	page 3-577 (253666-048US/Sep.2013)
vpabs(w b d)	PABSB PABSW PABSD PABSQ	Packed Absolute Value	page 5-404 (319433-016/Oct. 2013)
vpackss(dw wb)	PACKSSWB PACKSSDW	Pack with Signed Saturation	page 4-27 (253667-048US/Sep.2013)
vpackusdw	PACKUSDW	Pack with Unsigned Saturation	page 4-32 (253667-048US/Sep.2013)
vpackuswb	PACKUSWB	Pack with Unsigned Saturation	page 4-35 (253667-048US/Sep.2013)
vpadd(q w b d)	PADDB PADDW PADDD PADDQ	Add Packed Integers	page 5-408 (319433-016/Oct. 2013)
vpadds(w b)	PADDSB PADDSW	Add Packed Signed Integers with Signed Saturation	page 4-44 (253667-048US/Sep.2013)
vpaddus(w b)	PADDUSB PADDUSW	Add Packed Unsigned Integers with Unsigned Saturation	page 4-47 (253667-048US/Sep.2013)
vpalignr	PALIGNR	Packed Align Right	page 4-50 (253667-048US/Sep.2013)
vpand	PAND	Logical AND	page 5-413 (319433-016/Oct. 2013)
vpandn	PANDN	Logical AND NOT	page 5-416 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vpavg(w b)	PAVGB PAVGW	Average Packed Integers	page 4-58 (253667-048US/ Sep.2013)
vpblendvb	PBLENDVB	Variable Blend Packed Bytes	page 4-61 (253667-048US/ Sep.2013)
vpblendw	PBLENDW	Blend Packed Words	page 4-65 (253667-048US/ Sep.2013)
vpcmpsq(q w b d)	PCMPSEQ PCMPSEQW PCMPSEQD PCMPSEQQ	Compare Packed Integers for Equality	page 5-419 (319433-016/Oct. 2013)
vpcmpgt(q w b d)	PCMPGTB PCMPGTW PCMPGTD PCMPGTQ	Compare Packed Integers for Greater Than	page 5-424 (319433-016/Oct. 2013)
vphaddsw	PHADDSW	Packed Horizontal Add and Saturate	page 4-105 (253667-048US/ Sep.2013)
vphadd(w d)	PHADDW PHADDD	Packed Horizontal Add	page 4-101 (253667-048US/ Sep.2013)
vphsubsw	PHSUBSW	Packed Horizontal Subtract and Saturate	page 4-112 (253667-048US/ Sep.2013)
vphsub(w d)	PHSUBW PHSUBD	Packed Horizontal Subtract	page 4-109 (253667-048US/ Sep.2013)
vpaddubsw	PMADDUBSW	Multiply and Add Packed Signed and Unsigned Bytes	page 4-118 (253667-048US/ Sep.2013)
vpaddwd	PMADDWD	Multiply and Add Packed Integers	page 4-120 (253667-048US/ Sep.2013)
vpmaxs(w b d)	PMAXS PMAXS PMAXS	Maximum of Packed Signed Integers	page 5-471 (319433-016/Oct. 2013)

### 3.8 AVX2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	PMAXSQ		
vpmmaxub	PMAXUB	Maximum of Packed Unsigned Byte Integers	page 4-131 (253667-048US/Sep.2013)
vpmmaxud	PMAXUD PMAXUQ	Maximum of Packed Unsigned Integers	page 5-476 (319433-016/Oct. 2013)
vpmmaxuw	PMAXUW	Maximum of Packed Word Integers	page 4-136 (253667-048US/Sep.2013)
vpmminsb	PMINSB	Minimum of Packed Signed Byte Integers	page 4-138 (253667-048US/Sep.2013)
vpmminsd	PMINSD PMINSQ	Minimum of Packed Signed Integers	page 5-479 (319433-016/Oct. 2013)
vpmminsw	PMINSW	Minimum of Packed Signed Word Integers	page 4-143 (253667-048US/Sep.2013)
vpmminub	PMINUB	Minimum of Packed Unsigned Byte Integers	page 4-146 (253667-048US/Sep.2013)
vpmminud	PMINUD PMINUQ	Minimum of Packed Unsigned Integers	page 5-482 (319433-016/Oct. 2013)
vpmminuw	PMINUW	Minimum of Packed Word Integers	page 4-151 (253667-048US/Sep.2013)
vpmovmskb	PMOVMskb	Move Byte Mask	page 4-153 (253667-048US/Sep.2013)
vpmovsx (bd   bq   bw   dq   wd   wq)	PMOVsx	Packed Move with Sign Extend	page 5-500 (319433-016/Oct. 2013)
vpmovzx (bd   bq   bw   dq   wd   wq)	PMOVzX	Packed Move with Zero Extend	page 5-507 (319433-016/Oct. 2013)
vpmuldq	PMULDQ	Multiply Packed Doubleword Integers	page 5-514 (319433-016/Oct. 2013)
vpmulhrsw	PMULHRsw	Packed Multiply High with Round and Scale	page 4-165 (253667-048US/Sep.2013)
vpmulhuw	PMULHUw	Multiply Packed Unsigned Integers and	page 4-168 (253667-048US/Sep.2013)



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
		Store High Result	
vpmulhw	PMULHW	Multiply Packed Signed Integers and Store High Result	page 4-172 (253667-048US/Sep.2013)
vpmulld	PMULLD	Multiply Packed Integers and Store Low Result	page 5-516 (319433-016/Oct. 2013)
vpmullw	PMULLW	Multiply Packed Signed Integers and Store Low Result	page 4-177 (253667-048US/Sep.2013)
vpmuludq	PMULUDQ	Multiply Packed Unsigned Doubleword Integers	page 5-519 (319433-016/Oct. 2013)
vpor	POR	Bitwise Logical Or	page 5-521 (319433-016/Oct. 2013)
vpsadbw	PSADBW	Compute Sum of Absolute Differences	page 4-198 (253667-048US/Sep.2013)
vpsshufb	PSHUFB	Packed Shuffle Bytes	page 4-201 (253667-048US/Sep.2013)
vpsshufd	PSHUFD	Shuffle Packed Doublewords	page 5-533 (319433-016/Oct. 2013)
vpsshufhw	PSHUFW	Shuffle Packed High Words	page 4-206 (253667-048US/Sep.2013)
vpsshufw	PSHUFLW	Shuffle Packed Low Words	page 4-208 (253667-048US/Sep.2013)
vpsign(w b d)	PSIGNB PSIGNW PSIGND	Packed SIGN	page 4-211 (253667-048US/Sep.2013)
vpslldq	PSLLDQ	Shift Double Quadword Left Logical	page 4-215 (253667-048US/Sep.2013)
vpsll(q w d)	PSLLW PSLLD	Bit Shift Left	page 5-536 (319433-016/Oct. 2013)

### 3.8 AVX2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	PSLLQ		
vpsra(w d)	PSRAW PSRAD PSRAQ	Bit Shift Arithmetic Right	page 5-544 (319433-016/Oct. 2013)
vpsrldq	PSRLDQ	Shift Double Quadword Right Logical	page 4-228 (253667-048US/ Sep.2013)
vpsrl(q w d)	PSRLW PSRLD PSRLQ	Shift Packed Data Right Logical	page 5-550 (319433-016/Oct. 2013)
vpsub(q w b d)	PSUBB PSUBW PSUBD PSUBQ	Packed Integer Subtract	page 5-563 (319433-016/Oct. 2013)
vpsubs(w b)	PSUBSB PSUBSW	Subtract Packed Signed Integers with Signed Saturation	page 4-243 (253667-048US/ Sep.2013)
vpsubus(w b)	PSUBUSB PSUBUSW	Subtract Packed Unsigned Integers with Unsigned Saturation	page 4-246 (253667-048US/ Sep.2013)
vpunpckh(bw dq qdq wd)	PUNPCKHBW PUNPCKHWD PUNPCKHDQ PUNPCKHQDQ	Unpack High Data	page 5-571 (319433-016/Oct. 2013)
vpunpckl(bw dq qdq wd)	PUNPCKLBW PUNPCKLWD PUNPCKLDQ PUNPCKLQDQ	Unpack Low Data	page 5-578 (319433-016/Oct. 2013)
vpxor	PXOR PXORD PXORQ	Exclusive Or	page 5-612 (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vbroadcast(sd ss)	VBROADCAST	Load with Broadcast Floating-Point Data	page 5-27 (319433-016/Oct. 2013)
vextracti128	VEXTRACTI128 VEXTRACTI32x4 VEXTRACTI64x4	Extract packed Integer Values	page 5-155 (319433-016/Oct. 2013)
vgatherdp(d s)	VGATHERDPS VGATHERDPD	Gather Packed Single, Packed Double with Signed Dword	page 5-273 (319433-016/Oct. 2013)
vgatherqp(d s)	VGATHERQPS VGATHERQPD	Gather Packed Single, Packed Double with Signed Qword Indices	page 5-275 (319433-016/Oct. 2013)
vinerti128	VINSERTI128 VINSERTI32x4 VINSERTI64x4	Insert Packed Integer Values	page 5-308 (319433-016/Oct. 2013)
vpblendd	VPBLEND	Blend Packed Dwords	page 4-509 (253667-048US/Sep.2013)
vpbroadcast(q w b d)	VPBROADCAST	Load Integer and Broadcast	page 5-34 (319433-016/Oct. 2013)
vbroadcasti128	VPBROADCAST VBROADCASTI128	Broadcast Integer Data	page 4-511 (253667-048US/Sep.2013)
vperm2i128	VPERM2I128	Permute Integer Values	page 4-519 (253667-048US/Sep.2013)
vpermd	VPERMD	Permute Packed Doublewords/Elements	page 5-437 (319433-016/Oct. 2013)
vpermpd	VPERMPD	Permute Double-Precision Floating-Point Elements	page 5-455 (319433-016/Oct. 2013)
vpermps	VPERMPS	Permute Single-Precision Floating-Point Elements	page 5-458 (319433-016/Oct. 2013)

### 3.8 AVX2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vpermq	VPERMQ	Qwords Element Permutation	page 5-460 (319433-016/Oct. 2013)
vpgatherdd	VPGATHERDD VPGATHERDQ	Gather Packed Dword, Packed Qword with Signed Dword Indices	page 5-467 (319433-016/Oct. 2013)
vpgatherdq	VPGATHERDD VPGATHERDQ	Gather Packed Dword, Packed Qword with Signed Dword Indices	page 5-467 (319433-016/Oct. 2013)
vpgatherqd	VPGATHERQD VPGATHERQQ	Gather Packed Dword, Packed Qword with Signed Qword Indices	page 5-469 (319433-016/Oct. 2013)
vpgatherqq	VPGATHERQD VPGATHERQQ	Gather Packed Dword, Packed Qword with Signed Qword Indices	page 5-469 (319433-016/Oct. 2013)
vpmaskmov(q d)	VPMASKMOV	Conditional SIMD Integer Packed Loads and Stores	page 4-529 (253667-048US/Sep.2013)
vpsllv(q d)	VPSLLVW VPSLLVD VPSLLVQ	Variable Bit Shift Left Logical	page 5-557 (319433-016/Oct. 2013)
vpsravd	VPSRAVD VPSRAVQ	Variable Bit Shift Right Arithmetic	page 5-609 (319433-016/Oct. 2013)
vpsrlv(q d)	VPSRLVW VPSRLVD VPSRLVQ	Variable Bit Shift Right Logical	page 5-560 (319433-016/Oct. 2013)

## 3.9 BMI1 Instructions

**TABLE 26** BMI1 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
andn	ANDN	Logical AND NOT	page 3-53 (253666-048US/ Sep.2013)
bextr	BEXTR	Bit Field Extract	page 3-66 (253666-048US/ Sep.2013)
bsli	BLSI	Extract Lowest Set Isolated Bit	page 3-75 (253666-048US/ Sep.2013)
blsmk	BLSMSK	Get Mask Up to Lowest Set Bit	page 3-76 (253666-048US/ Sep.2013)
blsr	BLSR	Reset Lowest Set Bit	page 3-77 (253666-048US/ Sep.2013)
lzcnt( q l w)	LZCNT	Count the Number of Leading Zero Bits	page 3-476 (253666-048US/ Sep.2013)
tzcnt	TZCNT	Count the Number of Trailing Zero Bits	page 4-408 (253667-048US/ Sep.2013)

## 3.10 BMI2 Instructions

**TABLE 27** BMI2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
bzhi	BZHI	Zero High Bits Starting with Specified Bit Position	page 3-93 (253666-048US/ Sep.2013)
mulx	MULX	Unsigned Multiply Without Affecting Flags	page 3-593 (253666-048US/Sep.2013)
pdep	PDEP	Parallel Bits Deposit	page 4-91 (253667-048US/ Sep.2013)
pext	PEXT	Parallel Bits Extract	page 4-93 (253667-048US/ Sep.2013)
rorx	RORX	Rotate Right Logical Without Affecting Flags	page 4-311 (253667-048US/Sep.2013)
sarx	SARX SHLX SHRX	Shift Without Affecting Flags	page 4-335 (253667-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
shlx	SARX	Shift Without Affecting Flags	page 4-335 (253667-048US/Sep.2013)
	SHLX		
	SHRX		
shrx	SARX	Shift Without Affecting Flags	page 4-335 (253667-048US/Sep.2013)
	SHLX		
	SHRX		

## 3.11 F16C Instructions

**TABLE 28** F16C Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vcvtp2ps	VCVTPH2PS	Convert 16-bit FP values to Single-Precision FP values	page 5-93 (319433-016/Oct.2013)
vcvtps2ph	VCVTPS2PH	Convert Single-Precision FP value to 16-bit FP value	page 5-96 (319433-016/Oct.2013)

## 3.12 FMA Instructions

**TABLE 29** FMA Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
vfmadd132pd	VFMADD132PD	Fused Multiply-Add of Packed Double-Precision Floating-Point Values	page 4-436
	VFMADD213PD		
	VFMADD231PD		
vfmadd213pd	VFMADD132PD	Fused Multiply-Add of Packed Double-Precision Floating-Point Values	page 4-436
	VFMADD213PD		
	VFMADD231PD		
vfmadd231pd	VFMADD132PD	Fused Multiply-Add of Packed Double-Precision Floating-Point Values	page 4-436
	VFMADD213PD		

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VFMADD231PD		
vmadd132ps	VFMADD132PS VFMADD213PS VFMADD231PS	Fused Multiply-Add of Packed Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd213ps	VFMADD132PS VFMADD213PS VFMADD231PS	Fused Multiply-Add of Packed Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd231ps	VFMADD132PS VFMADD213PS VFMADD231PS	Fused Multiply-Add of Packed Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd132sd	VFMADD132SD VFMADD213SD VFMADD231SD	Fused Multiply-Add of Scalar Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd213sd	VFMADD132SD VFMADD213SD VFMADD231SD	Fused Multiply-Add of Scalar Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd231sd	VFMADD132SD VFMADD213SD VFMADD231SD	Fused Multiply-Add of Scalar Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd132ss	VFMADD132SS VFMADD213SS VFMADD231SS	Fused Multiply-Add of Scalar Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd213ss	VFMADD132SS VFMADD213SS VFMADD231SS	Fused Multiply-Add of Scalar Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vmadd231ss	VFMADD132SS VFMADD213SS VFMADD231SS	Fused Multiply-Add of Scalar Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vmaddsub132pd	VFMADDSUB132PD VFMADDSUB213PD	Fused Multiply-Alternating Add/Subtract of Packed	page (319433-016/Oct. 2013)

### 3.12 FMA Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VFMADDSUB231PD		
vmaddsub213pd	VFMADDSUB132PD VFMADDSUB213PD VFMADDSUB231PD	Fused Multiply-Alternating Add/Subtract of Packed	page (319433-016/Oct. 2013)
vmaddsub231pd	VFMADDSUB132PD VFMADDSUB213PD VFMADDSUB231PD	Fused Multiply-Alternating Add/Subtract of Packed	page (319433-016/Oct. 2013)
vmaddsub132ps	VFMADDSUB132PS VFMADDSUB213PS VFMADDSUB231PS	Fused Multiply-Alternating Add/Subtract of Packed	page (319433-016/Oct. 2013)
vmaddsub213ps	VFMADDSUB132PS VFMADDSUB213PS VFMADDSUB231PS	Fused Multiply-Alternating Add/Subtract of Packed	page (319433-016/Oct. 2013)
vmaddsub231ps	VFMADDSUB132PS VFMADDSUB213PS VFMADDSUB231PS	Fused Multiply-Alternating Add/Subtract of Packed	page (319433-016/Oct. 2013)
vfmsub132pd	VFMSUB132PD VFMSUB213PD VFMSUB231PD	Fused Multiply-Subtract of Packed Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub213pd	VFMSUB132PD VFMSUB213PD VFMSUB231PD	Fused Multiply-Subtract of Packed Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub231pd	VFMSUB132PD VFMSUB213PD VFMSUB231PD	Fused Multiply-Subtract of Packed Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub132ps	VFMSUB132PS VFMSUB213PS VFMSUB231PS	Fused Multiply-Subtract of Packed Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub213ps	VFMSUB132PS VFMSUB213PS	Fused Multiply-Subtract of Packed Single-Precision Floating-Point	page (319433-016/Oct. 2013)



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VFMSUB231PS		
vfmsub231ps	VFMSUB132PS VFMSUB213PS VFMSUB231PS	Fused Multiply-Subtract of Packed Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub132sd	VFMSUB132SD VFMSUB213SD VFMSUB231SD	Fused Multiply-Subtract of Scalar Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub213sd	VFMSUB132SD VFMSUB213SD VFMSUB231SD	Fused Multiply-Subtract of Scalar Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub231sd	VFMSUB132SD VFMSUB213SD VFMSUB231SD	Fused Multiply-Subtract of Scalar Double-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub132ss	VFMSUB132SS VFMSUB213SS VFMSUB231SS	Fused Multiply-Subtract of Scalar Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub213ss	VFMSUB132SS VFMSUB213SS VFMSUB231SS	Fused Multiply-Subtract of Scalar Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsub231ss	VFMSUB132SS VFMSUB213SS VFMSUB231SS	Fused Multiply-Subtract of Scalar Single-Precision Floating-Point	page (319433-016/Oct. 2013)
vfmsubadd132pd	VFMSUBADD132PD VFMSUBADD213PD VFMSUBADD231PD	Fused Multiply-Alternating Subtract/Add of Packed	page (319433-016/Oct. 2013)
vfmsubadd213pd	VFMSUBADD132PD VFMSUBADD213PD VFMSUBADD231PD	Fused Multiply-Alternating Subtract/Add of Packed	page (319433-016/Oct. 2013)
vfmsubadd231pd	VFMSUBADD132PD VFMSUBADD213PD	Fused Multiply-Alternating Subtract/Add of Packed	page (319433-016/Oct. 2013)

### 3.12 FMA Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VFMSUBADD231PD		
vfmsubadd132ps	VFMSUBADD132PS VFMSUBADD213PS VFMSUBADD231PS	Fused Multiply-Alternating Subtract/Add of Packed	page (319433-016/Oct. 2013)
vfmsubadd213ps	VFMSUBADD132PS VFMSUBADD213PS VFMSUBADD231PS	Fused Multiply-Alternating Subtract/Add of Packed	page (319433-016/Oct. 2013)
vfmsubadd231ps	VFMSUBADD132PS VFMSUBADD213PS VFMSUBADD231PS	Fused Multiply-Alternating Subtract/Add of Packed	page (319433-016/Oct. 2013)
vfnmadd132pd	VFNMADD132PD VFNMADD213PD VFNMADD231PD	Fused Negative Multiply-Add of Packed Double-Precision	page (319433-016/Oct. 2013)
vfnmadd213pd	VFNMADD132PD VFNMADD213PD VFNMADD231PD	Fused Negative Multiply-Add of Packed Double-Precision	page (319433-016/Oct. 2013)
vfnmadd231pd	VFNMADD132PD VFNMADD213PD VFNMADD231PD	Fused Negative Multiply-Add of Packed Double-Precision	page (319433-016/Oct. 2013)
vfnmadd132ps	VFNMADD132PS VFNMADD213PS VFNMADD231PS	Fused Negative Multiply-Add of Packed Single-Precision	page (319433-016/Oct. 2013)
vfnmadd213ps	VFNMADD132PS VFNMADD213PS VFNMADD231PS	Fused Negative Multiply-Add of Packed Single-Precision	page (319433-016/Oct. 2013)
vfnmadd231ps	VFNMADD132PS VFNMADD213PS VFNMADD231PS	Fused Negative Multiply-Add of Packed Single-Precision	page (319433-016/Oct. 2013)
vfnmadd132sd	VFNMADD132SD VFNMADD213SD	Fused Negative Multiply-Add of Scalar Double-Precision	page (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VFNMADD231SD		
vfnmadd213sd	VFNMADD132SD VFNMADD213SD VFNMADD231SD	Fused Negative Multiply-Add of Scalar Double-Precision	page (319433-016/Oct. 2013)
vfnmadd231sd	VFNMADD132SD VFNMADD213SD VFNMADD231SD	Fused Negative Multiply-Add of Scalar Double-Precision	page (319433-016/Oct. 2013)
vfnmadd132ss	VFNMADD132SS VFNMADD213SS VFNMADD231SS	Fused Negative Multiply-Add o	page 5-255(319433-016/Oct.2013)
vfnmadd213ss	VFNMADD132SS VFNMADD213SS VFNMADD231SS	Fused Negative Multiply-Add o	page 5-255(319433-016/Oct.2013)
vfnmadd231ss	VFNMADD132SS VFNMADD213SS VFNMADD231SS	Fused Negative Multiply-Add o	page 5-255(319433-016/Oct.2013)
vfnmsub132pd	VFNMSUB132PD VFNMSUB213PD VFNMSUB231PD	Fused Negative Multiply-Subtract of Packed Double- Precision Floating-Point Values	page 4-478(253667-048US/Sep.2013)
vfnmsub213pd	VFNMSUB132PD VFNMSUB213PD VFNMSUB231PD	Fused Negative Multiply-Subtract of Packed Double- Precision Floating-Point Values	page 4-478(253667-048US/Sep.2013)
vfnmsub231pd	VFNMSUB132PD VFNMSUB213PD VFNMSUB231PD	Fused Negative Multiply-Subtract of Packed Double- Precision Floating-Point Values	page 4-478(253667-048US/Sep.2013)
vfnmsub132ps	VFNMSUB132PS VFNMSUB213PS VFNMSUB231PS	Fused Negative Multiply-Subtract of Packed Single-Precision	page (319433-016/Oct. 2013)
vfnmsub213ps	VFNMSUB132PS VFNMSUB213PS	Fused Negative Multiply-Subtract of Packed Single-Precision	page (319433-016/Oct. 2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VFNMSUB231PS		
vfnmsub231ps	VFNMSUB132PS VFNMSUB213PS VFNMSUB231PS	Fused Negative Multiply-Subtract of Packed Single-Precision	page (319433-016/Oct. 2013)
vfnmsub132sd	VFNMSUB132SD VFNMSUB213SD VFNMSUB231SD	Fused Negative Multiply-Subtract of Scalar Double-Precision	page (319433-016/Oct. 2013)
vfnmsub213sd	VFNMSUB132SD VFNMSUB213SD VFNMSUB231SD	Fused Negative Multiply-Subtract of Scalar Double-Precision	page (319433-016/Oct. 2013)
vfnmsub231sd	VFNMSUB132SD VFNMSUB213SD VFNMSUB231SD	Fused Negative Multiply-Subtract of Scalar Double-Precision	page (319433-016/Oct. 2013)
vfnmsub132ss	VFNMSUB132SS VFNMSUB213SS VFNMSUB231SS	Fused Negative Multiply-Subtract of Scalar Single-Precision	page (319433-016/Oct. 2013)
vfnmsub213ss	VFNMSUB132SS VFNMSUB213SS VFNMSUB231SS	Fused Negative Multiply-Subtract of Scalar Single-Precision	page (319433-016/Oct. 2013)
vfnmsub231ss	VFNMSUB132SS VFNMSUB213SS VFNMSUB231SS	Fused Negative Multiply-Subtract of Scalar Single-Precision	page (319433-016/Oct. 2013)

## 3.13 FSGSBASE Instructions

**TABLE 30** FSGSBASE Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
rdfsbase( l q)	RDFSBASE RDGSBASE	Read FS/GS Segment Base	page 4-284 (253667-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
rdgsbase( l q)	RDFSBASE RDGSBASE	Read FS/GS Segment Base	page 4-284 (253667-048US/Sep.2013)
wrfsbase( l q)	WRFSBASE WRGSBASE	Write FS/GS Segment Base	page 4-548 (253667-048US/Sep.2013)
wrgsbase( l q)	WRFSBASE WRGSBASE	Write FS/GS Segment Base	page 4-548 (253667-048US/Sep.2013)

## 3.14 MMX Instructions

The MMX instructions enable x86 processors to perform single-instruction, multiple-data (SIMD) operations on packed byte, word, doubleword, or quadword integer operands contained in memory, in MMX registers, or in general-purpose registers.

### 3.14.1 Data Transfer Instructions (MMX)

The data transfer instructions move doubleword and quadword operands between MMX registers and between MMX registers and memory.

**TABLE 31** Data Transfer Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
movd	MOVD	move doubleword	movdq valid only under -m64
movq	MOVQ	move quadword	valid only under -m64

### 3.14.2 Conversion Instructions (MMX)

The conversion instructions pack and unpack bytes, words, and doublewords.

**TABLE 32** Conversion Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
packssdw	PACKSSDW	pack doublewords into words with signed saturation	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
packsswb	PACKSSWB	pack words into bytes with signed saturation	
packuswb	PACKUSWB	pack words into bytes with unsigned saturation	
punpckhbw	PUNPCKHBW	unpack high-order bytes	
punpckhdq	PUNPCKHDQ	unpack high-order doublewords	
punpckhwd	PUNPCKHWD	unpack high-order words	
punpcklbw	PUNPCKLBW	unpack low-order bytes	
punpckldq	PUNPCKLDQ	unpack low-order doublewords	
punpcklwd	PUNPCKLWD	unpack low-order words	

### 3.14.3 Packed Arithmetic Instructions (MMX)

The packed arithmetic instructions perform packed integer arithmetic on packed byte, word, and doubleword integers.

**TABLE 33** Packed Arithmetic Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
paddb	PADDB	add packed byte integers	
padd	PADD	add packed doubleword integers	
paddsb	PADDSB	add packed signed byte integers with signed saturation	
paddsw	PADDSW	add packed signed word integers with signed saturation	
paddusb	PADDUSB	add packed unsigned byte integers with unsigned saturation	
paddusw	PADDUSW	add packed unsigned word integers with unsigned saturation	
paddw	PADDW	add packed word integers	
pmaddwd	PMADDWD	multiply and add packed word integers	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
pmulhw	PMULHW	multiply packed signed word integers and store high result	
pmullw	PMULLW	multiply packed signed word integers and store low result	
psubb	PSUBB	subtract packed byte integers	
psubd	PSUBD	subtract packed doubleword integers	
psubsb	PSUBSB	subtract packed signed byte integers with signed saturation	
psubsw	PSUBSW	subtract packed signed word integers with signed saturation	
psubusb	PSUBUSB	subtract packed unsigned byte integers with unsigned saturation	
psubusw	PSUBUSW	subtract packed unsigned word integers with unsigned saturation	
psubw	PSUBW	subtract packed word integers	

### 3.14.4 Comparison Instructions (MMX)

The compare instructions compare packed bytes, words, or doublewords.

**TABLE 34** Comparison Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
pcmpeqb	PCMPEQB	compare packed bytes for equal	
pcmpeqd	PCMPEQD	compare packed doublewords for equal	
pcmpeqw	PCMPEQW	compare packed words for equal	
pcmpgtb	PCMPGTB	compare packed signed byte integers for greater than	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
pcmpgtd	PCMPGTD	compare packed signed doubleword integers for greater than	
pcmpgtw	PCMPGTW	compare packed signed word integers for greater than	

### 3.14.5 Logical Instructions (MMX)

The logical instructions perform logical operations on quadword operands.

**TABLE 35** Logical Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
pand	PAND	bitwise logical AND	
pandn	PANDN	bitwise logical AND NOT	
por	POR	bitwise logical OR	
pxor	PXOR	bitwise logical XOR	

### 3.14.6 Shift and Rotate Instructions (MMX)

The shift and rotate instructions operate on packed bytes, words, doublewords, or quadwords in 64-bit operands.

**TABLE 36** Shift and Rotate Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
pslld	PSLLD	shift packed doublewords left logical	
psllq	PSLLQ	shift packed quadword left logical	
psllw	PSLLW	shift packed words left logical	
psrad	PSRAD	shift packed doublewords right arithmetic	
psraw	PSRAW	shift packed words right arithmetic	
psrld	PSRLD	shift packed doublewords right logical	



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
psrlq	PSRLQ	shift packed quadword right logical	
psrlw	PSRLW	shift packed words right logical	

### 3.14.7 State Management Instructions (MMX)

The emms (EMMS) instruction clears the MMX state from the MMX registers.

**TABLE 37** State Management Instructions (MMX)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
emms	EMMS	empty MMX state	

## 3.15 MOVBE Instructions

**TABLE 38** MOVBE Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
movbe( q l w)	movbe	Reverse byte order in <source> and move to <destination>	325383-050US 3-519 Vol. 2A

## 3.16 PCLMULQDQ Instructions

**TABLE 39** PCLMULQDQ Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
pclmulqdq	PCLMULQDQ	Carry-Less Multiplication Quadword	page 4-68 (253667-048US/ Sep.2013)

## 3.17 PREFETCH Instructions

**TABLE 40** PREFETCH Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
prefetch	PREFETCH		page 256 (AMD:24594-Rev.3.20-May.2013)
prefetchw	PREFETCHW	Prefetch Data into Caches in Anticipation of a Write	page 872-873 (325383-053US/Jan.2015) page 256 (AMD:24594-Rev.3.20-May.2013)
prefetchwt1	PREFETCHWT1	Prefetch Vector Data Into Caches with Internet to Write and T1 Hint	page 874-875 (325383-053US/Jan.2015)

## 3.18 RDRAND Instructions

**TABLE 41** RDRAND Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
rdrand( q l w)	RDRAND	Returns a random number	page 10-1 (319433-016/Oct.2013)

## 3.19 RDSEED Instructions

**TABLE 42** RDSEED Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
rdseed( w l q)	RDSEED	Read Random SEED	page 971-972 (325383-053US/Jan.2015)

## 3.20 SSE Instructions

SSE instructions are an extension of the SIMD execution model introduced with the MMX technology. SSE instructions are divided into four subgroups:

- SIMD single-precision floating-point instructions that operate on the XMM registers
- MXSCR state management instructions
- 64-bit SIMD integer instructions that operate on the MMX registers
- Instructions that provide cache control, prefetch, and instruction ordering functionality

## 3.20.1 SIMD Single-Precision Floating-Point Instructions (SSE)

The SSE SIMD instructions operate on packed and scalar single-precision floating-point values located in the XMM registers or memory.

### 3.20.1.1 Data Transfer Instructions (SSE)

The SSE data transfer instructions move packed and scalar single-precision floating-point operands between XMM registers and between XMM registers and memory.

**TABLE 43** Data Transfer Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
movaps	MOVAPS	move four aligned packed single-precision floating-point values between XMM registers or memory	
movhlps	MOVHLPS	move two packed single-precision floating-point values from the high quadword of an XMM register to the low quadword of another XMM register	
movhps	MOVHPS	move two packed single-precision floating-point values to or from the high quadword of an XMM register or memory	
movlhps	MOVLHPS	move two packed single-precision floating-point values from the low quadword of an XMM register to the high quadword of another XMM register	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
movlps	MOVLPS	move two packed single-precision floating-point values to or from the low quadword of an XMM register or memory	
movmskps	MOVMSKPS	extract sign mask from four packed single-precision floating-point values	
movss	MOVSS	move scalar single-precision floating-point value between XMM registers or memory	
movups	MOVUPS	move four unaligned packed single-precision floating-point values between XMM registers or memory	

### 3.20.1.2 Packed Arithmetic Instructions (SSE)

SSE packed arithmetic instructions perform packed and scalar arithmetic operations on packed and scalar single-precision floating-point operands.

**TABLE 44** Packed Arithmetic Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
addps	ADDPS	add packed single-precision floating-point values	
addss	ADDSS	add scalar single-precision floating-point values	
divps	DIVPS	divide packed single-precision floating-point values	
divss	DIVSS	divide scalar single-precision floating-point values	
maxps	MAXPS	return maximum packed single-precision floating-point values	
maxss	MAXSS	return maximum scalar single-precision floating-point values	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
minps	MINPS	return minimum packed single-precision floating-point values	
minss	MINSS	return minimum scalar single-precision floating-point values.	
mulps	MULPS	multiply packed single-precision floating-point values	
mulss	MULSS	multiply scalar single-precision floating-point values	
rcpps	RCPPS	compute reciprocals of packed single-precision floating-point values	
rcpss	RCPSS	compute reciprocal of scalar single-precision floating-point values	
rsqrtps	RSQRTPS	compute reciprocals of square roots of packed single-precision floating-point values	
rsqrtss	RSQRTSS	compute reciprocal of square root of scalar single-precision floating-point values	
sqrtps	SQRTPS	compute square roots of packed single-precision floating-point values	
sqrtss	SQRTSS	compute square root of scalar single-precision floating-point values	
subps	SUBPS	subtract packed single-precision floating-point values	
subss	SUBSS	subtract scalar single-precision floating-point values	

### 3.20.1.3 Comparison Instructions (SSE)

The SSE compare instructions compare packed and scalar single-precision floating-point operands.

**TABLE 45** Comparison Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cmpps	CMPPS	compare packed single-precision floating-point values	
cmpss	CMPS	compare scalar single-precision floating-point values	
comiss	COMISS	perform ordered comparison of scalar single-precision floating-point values and set flags in EFLAGS register	
ucomiss	UCOMISS	perform unordered comparison of scalar single-precision floating-point values and set flags in EFLAGS register	

### 3.20.1.4 Logical Instructions (SSE)

The SSE logical instructions perform bitwise AND, AND NOT, OR, and XOR operations on packed single-precision floating-point operands.

**TABLE 46** Logical Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
andnps	ANDNPS	perform bitwise logical AND NOT of packed single-precision floating-point values	
andps	ANDPS	perform bitwise logical AND of packed single-precision floating-point values	
orps	ORPS	perform bitwise logical OR of packed single-precision floating-point values	
xorps	XORPS	perform bitwise logical XOR of packed single-precision floating-point values	

### 3.20.1.5 Shuffle and Unpack Instructions (SSE)

The SSE shuffle and unpack instructions shuffle or interleave single-precision floating-point values in packed single-precision floating-point operands.

**TABLE 47** Shuffle and Unpack Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
shufps	SHUFPS	shuffles values in packed single-precision floating-point operands	
unpckhps	UNPCKHPS	unpacks and interleaves the two high-order values from two single-precision floating-point operands	
unpcklps	UNPCKLPS	unpacks and interleaves the two low-order values from two single-precision floating-point operands	

### 3.20.1.6 Conversion Instructions (SSE)

The SSE conversion instructions convert packed and individual doubleword integers into packed and scalar single-precision floating-point values.

**TABLE 48** Conversion Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cvtpi2ps	CVTPI2PS	convert packed doubleword integers to packed single-precision floating-point values	
cvtps2pi	CVTPS2PI	convert packed single-precision floating-point values to packed doubleword integers	
cvtsi2ss	CVTSI2SS	convert doubleword integer to scalar single-precision floating-point value	
cvts2si	CVTS2SI	convert scalar single-precision floating-point value to a doubleword integer	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>cvttps2pi</code>	CVTTPS2PI	convert with truncation packed single-precision floating-point values to packed doubleword integers	
<code>cvttss2si</code>	CVTTSS2SI	convert with truncation scalar single-precision floating-point value to scalar doubleword integer	

## 3.20.2 MXCSR State Management Instructions (SSE)

The MXCSR state management instructions save and restore the state of the MXCSR control and status register.

**TABLE 49** MXCSR State Management Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>ldmxcsr</code>	LDMXCSR	load <code>%mxcsr</code> register	
<code>stmxcsr</code>	STMXCSR	save <code>%mxcsr</code> register state	

## 3.20.3 64-Bit SIMD Integer Instructions (SSE)

The SSE 64-bit SIMD integer instructions perform operations on packed bytes, words, or doublewords in MMX registers.

**TABLE 50** 64-Bit SIMD Integer Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
<code>pavgb</code>	PAVGB	compute average of packed unsigned byte integers	
<code>pavgw</code>	PAVGW	compute average of packed unsigned word integers	
<code>pextrw</code>	PEXTRW	extract word	
<code>pinsrw</code>	PINSRW	insert word	
<code>pmaxsw</code>	PMAXSW	maximum of packed signed word integers	
<code>pmaxub</code>	PMAXUB	maximum of packed unsigned byte integers	



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
pminsw	PMINSW	minimum of packed signed word integers	
pminub	PMINUB	minimum of packed unsigned byte integers	
pmovmskb	PMOVMSKB	move byte mask	
pmulhuw	PMULHUW	multiply packed unsigned integers and store high result	
psadbw	PSADBW	compute sum of absolute differences	
pshufw	PSHUFW	shuffle packed integer word in MMX register	

### 3.20.4 Miscellaneous Instructions (SSE)

The following instructions control caching, prefetching, and instruction ordering.

**TABLE 51** Miscellaneous Instructions (SSE)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
maskmovq	MASKMOVQ	non-temporal store of selected bytes from an MMX register into memory	
movntps	MOVNTPS	non-temporal store of four packed single-precision floating-point values from an XMM register into memory	
movntq	MOVNTQ	non-temporal store of quadword from an MMX register into memory	
prefetchnta	PREFETCHNTA	prefetch data into non-temporal cache structure and into a location close to the processor	
prefetcht0	PREFETCHT0	prefetch data into all levels of the cache hierarchy	
prefetcht1	PREFETCHT1	prefetch data into level 2 cache and higher	
prefetcht2	PREFETCHT2	prefetch data into level 2 cache and higher	
sfence	SFENCE	serialize store operations	

## 3.21 SSE2 Instructions

SSE2 instructions are an extension of the SIMD execution model introduced with the MMX technology and the SSE extensions. SSE2 instructions are divided into four subgroups:

- Packed and scalar double-precision floating-point instructions
- Packed single-precision floating-point conversion instructions
- 128-bit SIMD integer instructions
- Instructions that provide cache control and instruction ordering functionality

### 3.21.1 SSE2 Packed and Scalar Double-Precision Floating-Point Instructions

The SSE2 packed and scalar double-precision floating-point instructions operate on double-precision floating-point operands.

#### 3.21.1.1 SSE2 Data Movement Instructions

The SSE2 data movement instructions move double-precision floating-point data between XMM registers and memory.

**TABLE 52** SSE2 Data Movement Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
movapd	MOVAPD	move two aligned packed double-precision floating-point values between XMM registers and memory	
movhpd	MOVHPD	move high packed double-precision floating-point value to or from the high quadword of an XMM register and memory	
movlpd	MOVLPD	move low packed single-precision floating-point value to or from the low quadword of an XMM register and memory	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
movmskpd	MOVMSKPD	extract sign mask from two packed double-precision floating-point values	
movsd	MOVSD	move scalar double-precision floating-point value between XMM registers and memory.	
movupd	MOVUPD	move two unaligned packed double-precision floating-point values between XMM registers and memory	

### 3.21.1.2 SSE2 Packed Arithmetic Instructions

The SSE2 arithmetic instructions operate on packed and scalar double-precision floating-point operands.

**TABLE 53** SSE2 Packed Arithmetic Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
addpd	ADDPD	add packed double-precision floating-point values	
addsd	ADDSD	add scalar double-precision floating-point values	
divpd	DIVPD	divide packed double-precision floating-point values	
divsd	DIVSD	divide scalar double-precision floating-point values	
maxpd	MAXPD	return maximum packed double-precision floating-point values	
maxsd	MAXSD	return maximum scalar double-precision floating-point value	
minpd	MINPD	return minimum packed double-precision floating-point values	
minsd	MINSD	return minimum scalar double-precision floating-point value	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
mulpd	MULPD	multiply packed double-precision floating-point values	
mulsd	MULSD	multiply scalar double-precision floating-point values	
sqrtpd	SQRTPD	compute packed square roots of packed double-precision floating-point values	
sqrtsd	SQRSD	compute scalar square root of scalar double-precision floating-point value	
subpd	SUBPD	subtract packed double-precision floating-point values	
subsd	SUBSD	subtract scalar double-precision floating-point values	

### 3.21.1.3 SSE2 Logical Instructions

The SSE2 logical instructions operate on packed double-precision floating-point values.

**TABLE 54** SSE2 Logical Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
andnpd	ANDNPD	perform bitwise logical AND NOT of packed double-precision floating-point values	
andpd	ANDPD	perform bitwise logical AND of packed double-precision floating-point values	
orpd	ORPD	perform bitwise logical OR of packed double-precision floating-point values	
xorpd	XORPD	perform bitwise logical XOR of packed double-precision floating-point values	

### 3.21.1.4 SSE2 Compare Instructions

The SSE2 compare instructions compare packed and scalar double-precision floating-point values and return the results of the comparison to either the destination operand or to the EFLAGS register.

**TABLE 55** SSE2 Compare Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cmppd	CMPPD	compare packed double-precision floating-point values	
cmpsd	CMPSD	compare scalar double-precision floating-point values	
comisd	COMISD	perform ordered comparison of scalar double-precision floating-point values and set flags in EFLAGS register	
ucomisd	UCOMISD	perform unordered comparison of scalar double-precision floating-point values and set flags in EFLAGS register	

### 3.21.1.5 SSE2 Shuffle and Unpack Instructions

The SSE2 shuffle and unpack instructions operate on packed double-precision floating-point operands.

**TABLE 56** SSE2 Shuffle and Unpack Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
shufpd	SHUFPD	shuffle values in packed double-precision floating-point operands	
unpckhpd	UNPCKHPD	unpack and interleave the high values from two packed double-precision floating-point operands	
unpcklpd	UNPCKLPD	unpack and interleave the low values from two packed double-precision floating-point operands	

### 3.21.1.6 SSE2 Conversion Instructions

The SSE2 conversion instructions convert packed and individual doubleword integers into packed and scalar double-precision floating-point values (and vice versa). These instructions also convert between packed and scalar single-precision and double-precision floating-point values.

**TABLE 57** SSE2 Conversion Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cvtdq2pd	CVTDQ2PD	convert packed doubleword integers to packed double-precision floating-point values	
cvtpd2dq	CVTPD2DQ	convert packed double-precision floating-point values to packed doubleword integers	
cvtpd2pi	CVTPD2PI	convert packed double-precision floating-point values to packed doubleword integers	
cvtpd2ps	CVTPD2PS	convert packed double-precision floating-point values to packed single-precision floating-point values	
cvtpi2pd	CVTPI2PD	convert packed doubleword integers to packed double-precision floating-point values	
cvtps2pd	CVTPS2PD	convert packed single-precision floating-point values to packed double-precision floating-point values	
cvtsd2si	CVTSD2SI	convert scalar double-precision floating-point values to a doubleword integer	
cvtsd2ss	CVTSD2SS	convert scalar double-precision floating-point values to scalar single-precision floating-point values	
cvtsi2sd	CVTSI2SD	convert doubleword integer to scalar double-	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
		precision floating-point value	
cvtss2sd	CVTSS2SD	convert scalar single-precision floating-point values to scalar double-precision floating-point values	
cvttpd2dq	CVTTPD2DQ	convert with truncation packed double-precision floating-point values to packed doubleword integers	
cvttpd2pi	CVTTPD2PI	convert with truncation packed double-precision floating-point values to packed doubleword integers	
cvttsd2si	CVTTSD2SI	convert with truncation scalar double-precision floating-point values to scalar doubleword integers	

## 3.21.2 SSE2 Packed Single-Precision Floating-Point Instructions

The SSE2 packed single-precision floating-point instructions operate on single-precision floating-point and integer operands.

**TABLE 58** SSE2 Packed Single-Precision Floating-Point Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
cvtdq2ps	CVTDQ2PS	convert packed doubleword integers to packed single-precision floating-point values	
cvtps2dq	CVTPS2DQ	convert packed single-precision floating-point values to packed doubleword integers	
cvttps2dq	CVTTPS2DQ	convert with truncation packed single-precision floating-point values to packed doubleword integers	

### 3.21.3 SSE2 128-Bit SIMD Integer Instructions

The SSE2 SIMD integer instructions operate on packed words, doublewords, and quadwords contained in XMM and MMX registers.

**TABLE 59** SSE2 128-Bit SIMD Integer Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
movdq2q	MOVDQ2Q	move quadword integer from XMM to MMX registers	
movdqa	MOVDQA	move aligned double quadword	
movdqu	MOVDQU	move unaligned double quadword	
movq2dq	MOVQ2DQ	move quadword integer from MMX to XMM registers	
paddq	PADDQ	add packed quadword integers	
pmuludq	PMULUDQ	multiply packed unsigned doubleword integers	
pshufd	PSHUFQ	shuffle packed doublewords	
pshufhw	PSHUFHW	shuffle packed high words	
pshufwl	PSHUFWL	shuffle packed low words	
pslldq	PSLLDQ	shift double quadword left logical	
psrldq	PSRLDQ	shift double quadword right logical	
psubq	PSUBQ	subtract packed quadword integers	
punpckhqdq	PUNPCKHQDQ	unpack high quadwords	
punpcklqdq	PUNPCKLQDQ	unpack low quadwords	

### 3.21.4 SSE2 Miscellaneous Instructions

The SSE2 instructions described below provide additional functionality for caching non-temporal data when storing data from XMM registers to memory, and provide additional control of instruction ordering on store operations.



**TABLE 60** SSE2 Miscellaneous Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
clflush	CLFLUSH	flushes and invalidates a memory operand and its associated cache line from all levels of the processor's cache hierarchy	
lfence	LFENCE	serializes load operations	
maskmovdqu	MASKMOVDQU	non-temporal store of selected bytes from an XMM register into memory	
mfence	MFENCE	serializes load and store operations	
movntdq	MOVNTDQ	non-temporal store of double quadword from an XMM register into memory	
movnti	MOVNTI	non-temporal store of a doubleword from a general-purpose register into memory	movntiq valid only under -m64
movntpd	MOVNTPD	non-temporal store of two packed double-precision floating-point values from an XMM register into memory	
pause	PAUSE	improves the performance of spin-wait loops	

## 3.22 SSE3 Instructions

**TABLE 61** SSE3 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
addsubpd	ADDSUBPD	Packed Double-FP Add/Subtract	page 3-35 (253666-048US/Sep.2013)
addsubps	ADDSUBPS	Packed Single-FP Add/Subtract	page 3-37 (253666-048US/Sep.2013)
haddpd	HADDPD	Packed Double-FP Horizontal Add	page 3-370 (253666-048US/Sep.2013)
haddps	HADDPS	Packed Single-FP Horizontal Add	page 3-373 (253666-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
hsubpd	HSUBPD	Packed Double-FP Horizontal Subtract	page 3-377 (253666-048US/Sep.2013)
hsubps	HSUBPS	Packed Single-FP Horizontal Subtract	page 3-380 (253666-048US/Sep.2013)
lddqu	LDDQU	Load Unaligned Integer 128 Bits	page 3-444 (253666-048US/Sep.2013)
movddup	MOVDDUP	Replicate Double FP Values	page 5-346 (319433-016/Oct.2013)
movshdup	MOVSHDUP	Replicate Single FP Values	page 5-380 (319433-016/Oct.2013)
movsldup	MOVSLDUP	Replicate Single FP Values	page 5-383 (319433-016/Oct.2013)

## 3.23 SSE4a Instructions

**TABLE 62** SSE4a Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
extrq	EXTRQ		page 139 (AMD:26568-Rev.3.18-Oct.2013)
insertq	INSERTQ		page 154 (AMD:26568-Rev.3.18-Oct.2013)
movntsd	MOVNTSD		page 218 (AMD:26568-Rev.3.18-Oct.2013)
movntss	MOVNTSS		page 220 (AMD:26568-Rev.3.18-Oct.2013)

## 3.24 SSE4.1 Instructions

**TABLE 63** SSE4.1 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
blendpd	BLENDDP	Blend Packed Double Precision Floating-Point Values	page 3-64 (253666-048US/Sep.2013)
blendps	BLENDPS	Blend Packed Single Precision Floating-Point Values	page 3-68 (253666-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
blendvpd	BLENDVPD	Variable Blend Packed Double Precision Floating-Point Values	page 3-70 (253666-048US/Sep.2013)
blendvps	BLENDVPS	Variable Blend Packed Single Precision Floating-Point Values	page 3-72 (253666-048US/Sep.2013)
dppd	DPPD	Dot Product of Packed Double Precision Floating-Point Values	page 3-240 (253666-048US/Sep.2013)
dpps	DPPS	Dot Product of Packed Single Precision Floating-Point Values	page 3-242 (253666-048US/Sep.2013)
extractps	EXTRACTPS	Extract Packed Floating-Point Values	page 5-158 (319433-016/Oct.2013)
insertps	INSERTPS	Insert Scalar Single-Precision Floating-Point Value	page 5-311 (319433-016/Oct.2013)
movntdqa	MOVNTDQA	Load Double Quadword Non-Temporal Aligned Hint	page 5-369 (319433-016/Oct.2013)
mpsadbw	MPSADBW	Compute Multiple Packed Sums of Absolute Difference	page 3-577 (253666-048US/Sep.2013)
packusdw	PACKUSDW	Pack with Unsigned Saturation	page 4-32 (253667-048US/Sep.2013)
pblendvb	PBLENDVB	Variable Blend Packed Bytes	page 4-61 (253667-048US/Sep.2013)
pblendw	PBLENDW	Blend Packed Words	page 4-65 (253667-048US/Sep.2013)
pcmpqq	PCMPQEB PCMPQEW PCMPQEQD PCMPQEQQ	Compare Packed Integers for Equality	page 5-419 (319433-016/Oct.2013)
pextr(q b d)	PEXTRB PEXTRD PEXTRQ	Extract Byte/Dword/Qword	page 4-95 (253667-048US/Sep.2013)
pextrw	PEXTRW	Extract Word	page 4-98 (253667-048US/Sep.2013)
phminposuw	PHMINPOSUW	Packed Horizontal Word Minimum	page 4-107 (253667-048US/Sep.2013)

### 3.24 SSE4.1 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
pinsr(q b d)	PINSRB PINSRD PINSRQ	Insert Byte/Dword/Qword	page 4-114 (253667-048US/Sep.2013)
pmaxs(b d)	PMASB PMASW PMASD PMASQ	Maximum of Packed Signed Integers	page 5-471 (319433-016/Oct.2013)
pmaxud	PMAXUD PMAXUQ	Maximum of Packed Unsigned Integers	page 5-476 (319433-016/Oct.2013)
pmaxuw	PMAXUW	Maximum of Packed Word Integers	page 4-136 (253667-048US/Sep.2013)
pminsb	PMINSB	Minimum of Packed Signed Byte Integers	page 4-138 (253667-048US/Sep.2013)
pminsd	PMINSD PMINSQ	Minimum of Packed Signed Integers	page 5-479 (319433-016/Oct.2013)
pminud	PMINUD PMINUQ	Minimum of Packed Unsigned Integers	page 5-482 (319433-016/Oct.2013)
pminuw	PMINUW	Minimum of Packed Word Integers	page 4-151 (253667-048US/Sep.2013)
pmovsx(bd bq bw dq wd wq)	PMOVSX	Packed Move with Sign Extend	page 5-500 (319433-016/Oct.2013)
pmovzx(bd bq bw dq wd wq)	PMOVZX	Packed Move with Zero Extend	page 5-507 (319433-016/Oct.2013)
pmuldq	PMULDQ	Multiply Packed Doubleword Integers	page 5-514 (319433-016/Oct.2013)
pmulld	PMULLD	Multiply Packed Integers and Store Low Result	page 5-516 (319433-016/Oct.2013)
pptest	PTEST	Logical Compare	page 4-249 (253667-048US/Sep.2013)
roundpd	ROUNDPD	Round Packed Double Precision Floating-Point Values	page 4-312 (253667-048US/Sep.2013)
roundps	ROUNDPS	Round Packed Single Precision Floating-Point Values	page 4-315 (253667-048US/Sep.2013)
roundsd	ROUNDSD	Round Scalar Double Precision Floating-Point Values	page 4-318 (253667-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
roundss	ROUNDSS	Round Scalar Single Precision Floating-Point Values	page 4-320 (253667-048US/Sep.2013)

## 3.25 SSE4.2 Instructions

**TABLE 64** SSE4.2 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
pcmpestri	PCMPESTRI	Packed Compare Explicit Length Strings, Return Index	page 4-77 (253667-048US/Sep.2013)
pcmpestrm	PCMPESTRM	Packed Compare Explicit Length Strings, Return Mask	page 4-79 (253667-048US/Sep.2013)
pcmpgtq	PCMPGTB	Compare Packed Integers for Greater Than	page 5-424 (319433-016/Oct.2013)
pcmpistri	PCMPISTRI	Packed Compare Implicit Length Strings, Return Index	page 4-87 (253667-048US/Sep.2013)
pcmpistrm	PCMPISTRM	Packed Compare Implicit Length Strings, Return Mask	page 4-89 (253667-048US/Sep.2013)

## 3.26 SSSE3 Instructions

**TABLE 65** SSSE3 Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
pabs(w b d)	PABS PABSW PABSD PABSQ	Packed Absolute Value	page 5-404 (319433-016/Oct.2013)
palignr	PALIGNR	Packed Align Right	page 4-50 (253667-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
phaddsw	PHADDSW	Packed Horizontal Add and Saturate	page 4-105 (253667-048US/Sep.2013)
phadd(w d)	PHADDW PHADDD	Packed Horizontal Add	page 4-101 (253667-048US/Sep.2013)
phsubsw	PHSUBSW	Packed Horizontal Subtract and Saturate	page 4-112 (253667-048US/Sep.2013)
phsub(w d)	PHSUBW PHSUBD	Packed Horizontal Subtract	page 4-109 (253667-048US/Sep.2013)
pmaddubsw	PMADDUBSW	Multiply and Add Packed Signed and Unsigned Bytes	page 4-118 (253667-048US/Sep.2013)
pmulhrsw	PMULHRSW	Packed Multiply High with Round and Scale	page 4-165 (253667-048US/Sep.2013)
pshufb	PSHUFB	Packed Shuffle Bytes	page 4-201 (253667-048US/Sep.2013)
psign(w b d)	PSIGNB PSIGNW PSIGND	Packed SIGN	page 4-211 (253667-048US/Sep.2013)

## 3.27 Transactional Synchronization Extensions

**TABLE 66** HLE Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
xtest	XTEST	Test If In Transactional Execution	page 4-588 (253667-048US/Sep.2013)

**TABLE 67** RTM Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
xabort	XABORT	Transactional Abort	page 4-555 (253667-048US/Sep.2013)
xbegin( l w)	XBEGIN	Transactional Begin	page 4-559 (253667-048US/Sep.2013)
xend	XEND	Transactional End	page 4-564 (253667-048US/Sep.2013)

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
xtest	XTEST	Test If In Transactional Execution	page 4-588 (253667-048US/Sep.2013)

## 3.28 Operating System Support Instructions

The operating system support instructions provide functionality for process management, performance monitoring, debugging, and other systems tasks.

**TABLE 68** Operating System Support Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
arpl	ARPL	adjust requested privilege level	
clts	CLTS	clear the task-switched flag	
hlt	HLT	halt processor	
invd	INVD	invalidate cache, no writeback	
invlpg	INVLPG	invalidate TLB entry	
lar	LAR	load access rights	larq valid only under -m64
lgdt	LGDT	load global descriptor table (GDT) register	
lidt	LIDT	load interrupt descriptor table (IDT) register	
lldt	LLDT	load local descriptor table (LDT) register	
lmsw	LMSW	load machine status word	
lock	LOCK	lock bus	
lsl	LSL	load segment limit	lslq valid only under -m64
ltr	LTR	load task register	
rdmsr	RDMSR	read model-specific register	
rdpmc	RDPMC	read performance monitoring counters	
rdtsc	RDTSC	read time stamp counter	
rsm	RSM	return from system management mode (SMM)	
sgdt	SGDT	store global descriptor table (GDT) register	

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Notes
sidt	SIDT	store interrupt descriptor table (IDT) register	
sldt	SLDT	store local descriptor table (LDT) register	sldtq valid only under -m64
smsw	SMSW	store machine status word	smswq valid only under -m64
str	STR	store task register	strq valid only under -m64
sysenter	SYSENTER	fast system call, transfers to a flat protected model kernel at CPL=0	
sysexit	SYSEXIT	fast system call, transfers to a flat protected mode kernel at CPL=3	
verr	VERR	verify segment for reading	
verw	VERW	verify segment for writing	
wbinvd	WBINVD	invalidate cache, with writeback	
wrmsr	WRMSR	write model-specific register	

## 3.29 VMX Instructions

**TABLE 69** VMX Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
invept	INVEPT	Invalidate Translations Derived from EPT	page 30-3 (326019-048US/Sep.2013)
invvpid	INVVPID	Invalidate Translations Based on VPID	page 30-6 (326019-048US/Sep.2013)
vmcall	VMCALL	Call to VM Monitor	page 30-9 (326019-048US/Sep.2013)
vmclear	VMCLEAR	Clear Virtual-Machine Control Structure	page 30-11 (326019-048US/Sep.2013)
vmfunc	VMFUNC	Invoke VM function	page 30-13 (326019-048US/Sep.2013)
vmlaunch	VMLAUNCH VMRESUME	Launch/Resume Virtual Machine	page 30-14 (326019-048US/Sep.2013)
vmresume	VMLAUNCH	Launch/Resume Virtual Machine	page 30-14 (326019-048US/Sep.2013)



Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
	VMRESUME		
vmptlrd	VMPTRL	Load Pointer to Virtual-Machine Control Structure	page 30-17 (326019-048US/Sep.2013)
vmptrst	VMPTRST	Store Pointer to Virtual-Machine Control Structure	page 30-19 (326019-048US/Sep.2013)
vmread	VMREAD	Read Field from Virtual-Machine Control Structure	page 30-21 (326019-048US/Sep.2013)
vmwrite	VMWRITE	Write Field to Virtual-Machine Control Structure	page 0-24 (326019-048US/Sep.2013)
vmxoff	VMXOFF	Leave VMX Operation	page 30-27 (326019-048US/Sep.2013)
vmxon	VMXON	Enter VMX Operation	page 30-29 (326019-048US/Sep.2013)

### 3.30 XSAVE Instructions

**TABLE 70** XSAVE Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
xsaveopt(64 )	XSAVEOPT	Save Processor Extended States Optimized	page 4-583 (253667-048US/Sep.2013)

### 3.31 3DNow Instructions

**TABLE 71** 3DNow Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
femms	FEMMS		page 18 (AMD:26569-Rev.3.13-May.2013)
pavgusb	PAVGUSB		page 70 (AMD:26569-Rev.3.13-May.2013)
pf2id	PF2ID		page 88 (AMD:26569-Rev.3.13-May.2013)
pf2iw	PF2IW		page 90 (AMD:26569-Rev.3.13-May.2013)
pfacc	PFACC		page 92 (AMD:26569-Rev.3.13-May.2013)

### 3.31 3DNow Instructions

Oracle Solaris Mnemonic	Intel/AMD Mnemonic	Description	Reference
pfadd	PFADD		page 94 (AMD:26569-Rev.3.13-May.2013)
pfcmpeq	PFCMPEQ		page 96 (AMD:26569-Rev.3.13-May.2013)
pfcmpge	PFCMPGE		page 98 (AMD:26569-Rev.3.13-May.2013)
pfcmpgt	PFCMPGT		page 101 (AMD:26569-Rev.3.13-May.2013)
pfmax	PFMAX		page 103 (AMD:26569-Rev.3.13-May.2013)
pfmin	PFMIN		page 105 (AMD:26569-Rev.3.13-May.2013)
pfmul	PFMUL		page 107 (AMD:26569-Rev.3.13-May.2013)
pfnacc	PFNACC		page 109 (AMD:26569-Rev.3.13-May.2013)
pfpnacc	PFPNACC		page 112 (AMD:26569-Rev.3.13-May.2013)
pfrcp	PFRCP		page 115 (AMD:26569-Rev.3.13-May.2013)
pfrcpit1	PFRCPIT1		page 118 (AMD:26569-Rev.3.13-May.2013)
pfrcpit2	PFRCPIT2		page 121 (AMD:26569-Rev.3.13-May.2013)
pfrsqit1	PFRSQIT1		page 124 (AMD:26569-Rev.3.13-May.2013)
pfrsqrt	PFRSQRT		page 127 (AMD:26569-Rev.3.13-May.2013)
pfsub	PFSUB		page 130 (AMD:26569-Rev.3.13-May.2013)
pfsubr	PFSUBR		page 132 (AMD:26569-Rev.3.13-May.2013)
pi2fd	PI2FD		page 134 (AMD:26569-Rev.3.13-May.2013)
pi2fw	PI2FW		page 136 (AMD:26569-Rev.3.13-May.2013)
pmulhrw	PMULHRW		page 152 (AMD:26569-Rev.3.13-May.2013)
pswapd	PSWAPD		page 201 (AMD:26569-Rev.3.13-May.2013)

## 3.32 64-Bit AMD Opteron Considerations

To assemble code for the AMD Opteron CPU, invoke the assembler with the `-m64` command line option. See the [as\(1\)](#) man page for additional information.

The following Oracle Solaris mnemonics are only valid when the `-m64` command line option is specified:

adcq	cmovnoq	mulq
addq	cmovnpq	negq
andq	cmovnsq	notq
bsfq	cmovnzq	orq
bsrq	cmovoq	popfq
bswapq	cmovpeq	popq
btcq	cmovpoq	pushfq
btq	cmovpq	pushq
btrq	cmovsq	rclq
btsq	cmovzq	rcrq
cltq	cmpq	rolq
cmovaeq	cmpsq	rorq
cmovaq	cmpxchgq	salq
cmovbeq	cqtd	sarq
cmovbq	cqto	sbbq
cmovcq	decq	scasq
cmoveq	divq	shldq
cmovgeq	idivq	shlq
cmovgq	imulq	shrdq
cmovleq	incq	shrq
cmovlq	larq	sldtq
cmovnaeq	leaq	smswq
cmovnaq	lodsq	stosq
cmovnbeq	lslq	strq
cmovnbq	movabs	subq
cmovncq	movdq	testq
cmovneq	movntiq	xaddq
cmovngeq	movq	xchgq
cmovngq	movsq	xchgqA
cmovnlq	movswq	xorq
cmovnlq	movzsq	

The following Oracle Solaris mnemonics are *not* valid when the `-m64` command line option is specified:

aaa	daa	lesw
aad	das	popa
aam	into	popaw
aas	jecz	pusha
boundw	ldsw	pushaw

## Using the Assembler Command Line

---

This appendix describes how to invoke the assembler from the command line, and details the command-line options.

### A.1 Assembler Command Line

You invoke the assembler command line as follows:

```
as [options] [inputfile] ...
```

---

**Note** - The Oracle Solaris Studio C, C++, and Fortran compilers (`cc(1)`, `CC(1)`, and `f95(1)`) invoke the assembler with the `fbc` command. You can use either the `as` or `fbc` command on a Oracle Solaris platform to invoke the assembler. On an Oracle Solaris x86 platform, the `as` or `fbc` command will invoke the x86 assembler. On an Oracle Solaris SPARC platform, the command invokes the SPARC assembler.

---

The `as` command translates the assembly language source files, *inputfile*, into an executable object file, *objfile*. The assembler recognizes the filename argument *hyphen* (-) as the standard input. It accepts more than one file name on the command line. The input file is the concatenation of all the specified files. If an invalid option is given or the command line contains a syntax error, the assembler prints the error (including a synopsis of the command line syntax and options) to standard error output, and then terminates.

The assembler supports macros, `#include` files, and symbolic substitution through use of the C preprocessor `cpp(1)`. The assembler invokes the preprocessor before assembly begins if it has been specified from the command line as an option. (See the `-P` option.)

## A.2 Assembler Command Line Options

`-a32`

Allow 32-bit addresses in 64-bit mode.

`-Dname -Dname=def`

When the `-P` option is in effect, these options are passed to the `cpp` preprocessor without interpretation by the `as` command; otherwise, they are ignored.

`-{n}H`

Enable (`-H`) or suppress (`-nH`) generation of the Hardware Capabilities section.

`-Ipath`

When the `-P` option is in effect, this option is passed to the `cpp` preprocessor without interpretation by the `as` command; otherwise, it is ignored.

`-i`

Ignore line number information from the preprocessor.

`-KPIC`

Check for address referencing with absolute relocation and issue warning.

`-m`

This option runs `m4` macro preprocessing on input. The `m4` preprocessor is more useful for complex preprocessing than the C preprocessor invoked by the `-P` option. See the `m4(1)` man page for more information about the `m4` macro-processor.

`-m64 | -m32`

Select the 64-bit (`-m64`) or 32-bit (`-m32`) memory model. With `-m64`, the resulting `.o` object files are in 64-bit ELF format and can only be linked with other object files in the same

format. The resulting executable can only be run on a 64-bit x86 processor running 64-bit Oracle Solaris OS. `-m32` is the default.

`-n`

Suppress all warnings while assembling.

`-o outfile`

Write the output of the assembler to *outfile*. By default, if `-o` is not specified, the output file name is the same as the input file name with `.s` replaced with `.o`.

`-P`

Run `cpp(1)`, the C preprocessor, on the files being assembled. The preprocessor is run separately on each input file, not on their concatenation. The preprocessor output is passed to the assembler.

`-Q{y|n}`

This option produces the "assembler version" information in the comment section of the output object file if the `y` option is specified; if the `n` option is specified, the information is suppressed.

`-S[a|b|c|l|A|B|C|L]`

Produces a disassembly of the emitted code to the standard output. Adding each of the following characters to the `-S` option produces:

- `a` - disassembling with address
- `b` - disassembling with `".bof"`
- `c` - disassembling with comments
- `l` - disassembling with line numbers

Capital letters turn the switch off for the corresponding option.

`-s`

This option places all stabs in the `".stabs"` section. By default, stabs are placed in `".stabs.exc1"` sections, which are stripped out by the static linker `ld` during final execution. When

the `-s` option is used, stabs remain in the final executable because ".stab" sections are not stripped out by the static linker `ld`.

`-Uname`

When the `-P` option is in effect, this option is passed to the `cpp` preprocessor without interpretation by the `as` command; otherwise, it is ignored.

`-V`

This option writes the version information on the standard error output.

`-xchip=processor`

*processor* specifies the target architecture processor. When there is a choice between several possible encodings, choose the one that is appropriate for the stated chip. In particular, use the appropriate no-op byte sequence to fill code alignment padding, and warn when instructions not defined for the stated chip are used.

The assembler accepts the instruction sets for the following recognized `-xchip` processor values:

<i>processor</i> value	Target Processor
<code>generic</code>	Generic x86
<code>native</code>	Host processor.
<code>core2</code>	Intel Core2
<code>nehalem</code>	Intel Nehalem
<code>opteron</code>	AMD Opteron
<code>penryn</code>	Intel Penryn
<code>pentium</code>	Intel Pentium
<code>pentium_pro</code>	Intel Pentium Pro
<code>pentium3</code>	Intel Pentium 3.
<code>pentium4</code>	Intel Pentium 4
<code>sandybridge</code>	Intel Sandy Bridge
<code>westmere</code>	Intel Westmere
<code>amdfam10</code>	AMD FAM10
<code>ivybridge</code>	Intel Ivy Bridge
<code>haswell</code>	Intel Haswell
<code>broadwell</code>	Intel Broadwell



`-xmodel=[small | medium | kernel]`

For `-m64` only, generate `R_X86_64_32S` relocatable type for data access under `kernel`. Otherwise, generate `R_X86_64_32` under `small`. `SHN_AMD64_LCOMMON` and `.lbcomm` support added under `medium`. The default is `small`.

`-Y{d|m},path`

Specify the path to locate the version of `cm4defs` (`-Yd,path`) or `m4` (`-Ym,path`) to use.

`-YI,path`

Indicate path to search for `#include` header files.

## A.3 Disassembling Object Code

The `dis` program is the object code disassembler for ELF. It produces an assembly language listing of the object file. For detailed information about this function, see the `dis(1)` man page.



# Index

---

## A

- a32 option, 126
- aaa, 39
- aad, 39
- aam, 39
- aas, 39
- adc, 39
- adcx, 53
- add, 39
- addpd, 107
- addps, 100
- addressing, 23
- addsd, 107
- addss, 100
- addsubpd, 113
- addsubps, 113
- adox, 53
- aesdec, 53
- aesdeclast, 53
- aesenc, 53
- aesenclast, 53
- aesimc, 53
- aeskeygenassist, 53
- .align, 24
- and, 40
- andn, 85
- andnpd, 108
- andnps, 102
- andpd, 108
- andps, 102
- arpl, 119
- as

- syntax, UNIX versus Intel, 15
- as command, 125
- .ascii, 24
- assembler command line, 125
- assembler command line options, 126

## B

- .bcd, 24
- bextr, 85
- binary arithmetic instructions, 39
- bit instructions, 41
- blendpd, 114
- blendps, 114
- blendvpd, 115
- blendvps, 115
- blsi, 85
- blmsk, 85
- blsr, 85
- bound, 42
- bsf, 41
- bsr, 41
- .bss, 25
- bswap, 36
- bt, 41
- btc, 41
- btr, 41
- bts, 41
- .2byte, 25
- .4byte, 25
- .8byte, 25
- .byte, 25

byte instructions, 41

bzhi, 85

## C

.cfi\_adjust\_cfa\_offset OFFSET, 25  
.cfi\_def\_cfa REGISTER, OFFSET, 25  
.cfi\_def\_cfa\_offset OFFSET, 25  
.cfi\_def\_cfa\_register REGISTER, 25  
.cfi\_endproc, 25  
.cfi\_escape EXPRESSION[, ...], 26  
.cfi\_offset REGISTER, OFFSET, 26  
.cfi\_rel\_offset REGISTER, OFFSET, 26  
.cfi\_startproc, 27

call, 42

cbtw, 36

clc, 46

cld, 46

clflush, 113

cli, 46

cltd, 36

cltq, 36

clts, 119

cmc, 46

cmova, 36

cmovae, 36

cmov.ae, 36

cmovb, 36

cmov.b, 36

cmovbe, 36

cmov.be, 36

cmovc, 36

cmov.c, 36

cmove, 36

cmov.e, 36

cmovg, 36

cmov.g, 36

cmovge, 36

cmov.ge, 36

cmovl, 36

cmov.l, 36

cmovle, 36

cmov.le, 36

cmovna, 36

cmov.na, 36

cmovnae, 36

cmov.nae, 36

cmovnb, 36

cmov.nb, 36

cmovnbe, 37

cmov.nbe, 37

cmovnc, 37

cmov.nc, 37

cmovne, 37

cmov.ne, 37

cmovng, 37

cmov.ng, 37

cmovnge, 37

cmov.nge, 37

cmovnl, 37

cmov.nl, 37

cmovnle, 37

cmov.nle, 37

cmovno, 37

cmov.no, 37

cmovnp, 37

cmov.np, 37

cmovns, 37

cmov.ns, 37

cmovnz, 37

cmov.nz, 37

cmovo, 37

cmov.o, 37

cmovp, 37

cmov.p, 37

cmovpe, 37

cmovpo, 37

cmovs, 37

cmovz, 37

cmp, 39

cmppd, 109

cmpps, 102  
cmps, 44  
cmpsb, 44  
cmpsd, 109  
cmpsl, 44  
cmpss, 102  
cmpsw, 44  
cmpxchg, 37  
cmpxchg8b, 37  
comisd, 109  
comiss, 102  
.comm, 27  
command-line options, 126  
comment, 17  
compiler drivers, 125  
control transfer instructions, 42  
cpuid, 46  
cqtd, 37  
cqto, 38  
cvtdq2pd, 110  
cvtdq2ps, 111  
cvtpd2dq, 110  
cvtpd2pi, 110  
cvtpd2ps, 110  
cvtpi2pd, 110  
cvtpi2ps, 103  
cvtps2dq, 111  
cvtps2pd, 110  
cvtps2pi, 103  
cvtsd2si, 110  
cvtsd2ss, 110  
cvtsi2sd, 110  
cvtsi2ss, 103  
cvtss2sd, 111  
cvtss2si, 103  
cvttpd2dq, 111  
cvttpd2pi, 111  
cvttps2dq, 111  
cvttps2pi, 104  
cvttsd2si, 111  
cvtts2si, 104

cwtd, 38  
cwtl, 38

## D

-D option, 126  
daa, 40  
das, 40  
.data, 27  
data transfer instructions, 36  
dec, 39  
decimal arithmetic instructions, 39  
directives, 24  
dis program, 129  
disassembling object code, 129  
div, 39  
divpd, 107  
divps, 100  
divsd, 107  
divss, 100  
.double, 28  
dppd, 115  
dpps, 115

## E

emms, 97  
enter, 42  
.even, 28  
.ext, 28  
extensions  
    transactional synchronization, 118  
extractps, 115  
extrq, 114

## F

f2xm1, 50  
fabs, 48  
fadd, 48  
faddp, 48  
fbc command, 125

fbld, 47  
fbstp, 47  
fchs, 48  
fclex, 51  
fcmovb, 47  
fcmovbe, 47  
fcmovbe, 47  
fcmovnb, 47  
fcmovnbe, 47  
fcmovne, 47  
fcmovnu, 47  
fcmovu, 48  
fcom, 49  
fcomi, 49  
fcomip, 49  
fcomp, 49  
fcompp, 49  
fcos, 50  
fdecstp, 51  
fdiv, 48  
fdivp, 48  
fdivr, 48  
fdivrp, 48  
femms, 121  
ffree, 51  
fiadd, 48  
ficom, 49  
ficomp, 49  
fidiv, 48  
fidivr, 48  
fild, 48  
.file, 28  
fimul, 48  
fincstp, 51  
finit, 51  
fist, 48  
fistp, 48  
fisub, 48  
fisubr, 48  
flag control instructions, 45  
fld, 48  
fld1, 50  
fldcw, 51  
fldenv, 51  
fldl2e, 51  
fldl2t, 51  
fldlg2, 51  
fldln2, 51  
fldpi, 51  
fldz, 51  
.float, 28  
floating-point instructions  
    basic arithmetic, 48  
    comparison, 49  
    control, 51  
    data transfer, 47  
    load constants, 50  
    logarithmic *See* transcendental  
    transcendental, 50  
    trigonometric *See* transcendental  
fmul, 48  
fmulp, 49  
fnclex, 51  
fninit, 51  
fnop, 51  
fnsave, 52  
fnstcw, 52  
fnstenv, 52  
fnstsw, 52  
fpatan, 50  
fpem, 49  
fpem1, 49  
fptan, 50  
frndint, 49  
frstor, 52  
fsave, 52  
fscale, 49  
fsin, 50  
fsincos, 50  
fsqrt, 49  
fst, 48  
fstcw, 52

fstenv, 52  
 fstp, 48  
 fstsw, 52  
 fsub, 49  
 fsubp, 49  
 fsubr, 49  
 fsubrp, 49  
 ftst, 49  
 fucom, 49  
 fucomi, 50  
 fucomip, 50  
 fucomp, 50  
 fucompp, 50  
 fwait, 52  
 fxam, 50  
 fxch, 48  
 fxrstor, 52  
 fxsave, 52  
 fextract, 49  
 fyl2x, 50  
 fyl2xp1, 50

**G**

gas, 15  
 .globl, 28  
 .group, 28

**H**

-{n}H option, 126  
 haddpd, 113  
 haddps, 113  
 .hidden, 28  
 hlt, 119  
 hsubpd, 114  
 hsubps, 114  
 hyphen (-), 125

**I**

-I option, 126  
 -i option, 126  
 I/O (input/output) instructions, 45  
 .ident, 28  
 identifier, 19  
 idiv, 39  
 imul, 39  
 in, 45  
 inc, 39  
 ins, 45  
 insb, 45  
 insertps, 115  
 insertq, 114  
 insl, 45  
 instruction, 22  
   format, 22  
   suffixes, 22  
 instructions  
   3DNow, 121  
   Advanced Vector Extensions, 53  
   ADX, 53  
   AES, 53  
   AVX, 54  
   AVX2, 77  
   binary arithmetic, 39  
   bit, 41  
   BMI1, 85  
   BMI2, 85  
   byte, 41  
   control transfer, 42  
   data transfer, 36  
   decimal arithmetic, 39  
   F16C, 86  
   flag control, 45  
   floating-point, 47  
   FMA, 86  
   FSGSBASE, 92  
   I/O (input/output), 45  
   logical, 40  
   miscellaneous, 46  
   MMX, 93  
   MOVBE, 97

- operating system support, 119
- Opteron, 123
- PCLMULQDQ, 97
- PREFETCH, 98
- RDRAND, 98
- RDSEED, 98
- rotate, 40
- segment register, 46
- shift, 40
- SIMD state management, 52
- SSE, 98
- SSE2, 106
- SSE3, 113
- SSE4.1, 114
- SSE4.2, 117
- SSE4a, 114
- SSSE3, 117
- string, 44
- VMX, 120
- XSAVE, 121

- insw, 45
- int, 42
- into, 42
- invd, 119
- invept, 120
- invlpg, 119
- invoking, as command, 125
- invpcid, 38
- invvpid, 120
- iret, 42

**J**

- ja, 42
- jae, 42
- jb, 42
- jbe, 42
- jc, 42
- jcxz, 42
- je, 42
- jecxz, 42
- jpg, 42

- jge, 42
- jl, 43
- jle, 43
- jmp, 43
- jnae, 43
- jnb, 43
- jnbbe, 43
- jnc, 43
- jne, 43
- jng, 43
- jnge, 43
- jnl, 43
- jnle, 43
- jno, 43
- jnp, 43
- jns, 43
- jnz, 43
- jo, 43
- jp, 43
- jpe, 43
- jpo, 43
- js, 43
- jz, 43

**K**

- keyword, 19
- KPIC option, 126

**L**

- label, 18
  - numeric, 18
  - symbolic, 18
- lahf, 46
- lar, 119
- lcall, 43
- .lcomm, 29
- lddqu, 114
- ldmxcsr, 104
- lds, 46



lea, 46  
leave, 43  
les, 46  
lfence, 113  
lfs, 46  
lgdt, 119  
lgs, 46  
lidt, 119  
lldt, 119  
lmsw, 119  
.local, 29  
lock, 119  
lods, 44  
lodsb, 44  
lodsl, 44  
lodsw, 44  
logical instructions, 40  
.long, 29  
loop, 43  
loope, 43  
loopne, 43  
loopnz, 43  
loopz, 43  
lret, 43  
lsl, 119  
lss, 46  
ltr, 119  
lzcnt(|q|l|w), 85

## M

-m option, 126  
-m64 and -m32 options, 126  
maskmovdqu, 113  
maskmovq, 105  
maxpd, 107  
maxps, 100  
maxsd, 107  
maxss, 100  
mfence, 113  
minpd, 107  
minps, 101  
minsd, 107  
minss, 101  
miscellaneous instructions, 46  
MMX instructions  
    comparison, 95  
    conversion, 93  
    data transfer, 93  
    logical, 96  
    packed arithmetic, 94  
    rotate, 96  
    shift, 96  
    state management, 97  
mov, 38  
movabs, 38  
movabsA, 38  
movapd, 106  
movaps, 99  
movbe(|q|l|w), 97  
movd, 93  
movddup, 114  
movdq2q, 112  
movdqa, 112  
movdqu, 112  
movhlps, 99  
movhpd, 106  
movhps, 99  
movlhps, 99  
movlpd, 106  
movlps, 100  
movmskpd, 107  
movmskps, 100  
movntdq, 113  
movntdqa, 115  
movnti, 113  
movntpd, 113  
movntps, 105  
movntq, 105  
movntsd, 114  
movntss, 114  
movq, 93

movq2dq, 112  
movs, 44  
movsb, 38, 44  
movsd, 107  
movshdup, 114  
movsl, 44  
movsldup, 114  
movss, 100  
movsw, 38, 44  
movupd, 107  
movups, 100  
movzb, 38  
movzw, 38  
mpsadbw, 115  
mul, 39  
mulpd, 108  
mulps, 101  
mulsd, 108  
mulss, 101  
multiple files, on , 125  
mulx, 85

## N

-n option, 127  
neg, 39  
nop, 46  
not, 40  
numbers, 19  
    floating point, 20  
    integers, 20  
        binary, 20  
        decimal, 20  
        hexadecimal, 20  
        octal, 20

## O

-o option , 127  
operands, 23  
    immediate, 23  
    indirect, 23

memory  
    addressing, 23  
    ordering (source, destination), 23  
    register, 23  
operating system support instructions, 119  
Opteron instructions, 123  
options  
    command-line, 126  
or, 40  
orpd, 108  
orps, 102  
out, 45  
outs, 45  
outsb, 45  
outsl, 45  
outsw, 45

## P

-P option , 127  
pabs(w|b|d), 117  
packssdw, 93  
packsswb, 94  
packusdw, 115  
packuswb, 94  
paddb, 94  
padd, 94  
paddq, 112  
paddsb, 94  
paddsw, 94  
paddusb, 94  
paddusw, 94  
paddw, 94  
palignr, 117  
pand, 96  
pandn, 96  
pause, 113  
pavgb, 104  
pavgsb, 121  
pavgw, 104  
pblendvb, 115

---

pblendw, 115  
pclmulqdq, 97  
pcmpeqb, 95  
pcmpeqd, 95  
pcmpeq, 115  
pcmpeqw, 95  
pcmpetri, 117  
pcmpetrm, 117  
pcmpgtb, 95  
pcmpgtd, 96  
pcmpgtq, 117  
pcmpgtw, 96  
pcmpistri, 117  
pcmpistrm, 117  
pdep, 85  
pext, 85  
pextr(q|b|d), 115  
pextrw, 104, 115  
pf2id, 121  
pf2iw, 121  
pfacc, 121  
pfadd, 122  
pfcmeq, 122  
pfcmpge, 122  
pfcmpgt, 122  
pfmax, 122  
pfmin, 122  
pfmul, 122  
pfnacc, 122  
pfpnacc, 122  
pfrcp, 122  
pfrcpit1, 122  
pfrcpit2, 122  
pfrsqit1, 122  
pfrsqr, 122  
pfsb, 122  
pfsbr, 122  
phadd(w|d), 118  
phaddsw, 118  
phminposuw, 115  
phsub(w|d), 118  
phsubsw, 118  
pi2fd, 122  
pi2fw, 122  
pinsr(q|b|d), 116  
pinsrw, 104  
pmaddubsw, 118  
pmaddwd, 94  
pmaxs(b|d), 116  
pmaxsw, 104  
pmaxub, 104  
pmaxud, 116  
pmaxuw, 116  
pminsb, 116  
pminsd, 116  
pminsw, 105  
pminub, 105  
pminud, 116  
pminuw, 116  
pmovmskb, 105  
pmovsx(bd|bq|bw|dq|wd|wq), 116  
pmovzx(bd|bq|bw|dq|wd|wq), 116  
pmuldq, 116  
pmulhrsw, 118  
pmulhrw, 122  
pmulhw, 105  
pmulhw, 95  
pmulld, 116  
pmullw, 95  
pmuludq, 112  
pop, 38  
popa, 38  
popal, 38  
popaw, 38  
popf, 46  
popfw, 46  
.popsection, 29  
por, 96  
prefetch, 98  
prefetchnta, 105  
prefetcht0, 105  
prefetcht1, 105

prefetcht2, 105  
prefetchw, 98  
prefetchwt1, 98  
.previous, 29  
psadbw, 105  
pshufb, 118  
pshufd, 112  
pshufhw, 112  
pshuflw, 112  
pshufw, 105  
psign(w|b|d), 118  
pslld, 96  
pslldq, 112  
psllq, 96  
psllw, 96  
psrad, 96  
psraw, 96  
psrld, 96  
psrldq, 112  
psrlq, 96  
psrlw, 96  
psubb, 95  
psubd, 95  
psubq, 112  
psubsb, 95  
psubsw, 95  
psubusb, 95  
psubusw, 95  
psubw, 95  
pswapd, 122  
ptest, 116  
punpckhbw, 94  
punpckhdq, 94  
punpckhqdq, 112  
punpckhwd, 94  
punpcklbw, 94  
punpckldq, 94  
punpcklqdq, 112  
punpcklwd, 94  
push, 38  
pusha, 38

pushal, 38  
pushaw, 38  
pushf, 46  
pushfw, 46  
.pushsection, 29  
pxor, 96

## Q

-Q option, 127  
.quad, 29

## R

rcl, 40  
rcpps, 101  
rcps, 101  
rcr, 40  
rdfsbase(|l|q), 92  
rdgsbase(|l|q), 93  
rdmsr, 119  
rdpmc, 119  
rdrand(|q|l|w), 98  
rdseed(|w|l|q), 98  
rdtsc, 119  
.rel, 30  
rep, 44  
repnz, 44  
repz, 44  
ret, 43  
rol, 40  
ror, 40  
rorx, 85  
rotate instructions, 40  
roundpd, 116  
roundps, 116  
roundsd, 116  
roundss, 117  
rsm, 119  
rsqrtps, 101

rsqrtss, 101

## S

-S option , 127

-s option , 127

sahf, 46

sal, 40

sar, 40

sarx, 85

sbb, 39

scas, 44

scasb, 45

scasl, 45

scasw, 45

.section, 30

segment register instructions, 46

.set, 30

seta, 41

setae, 41

setb, 41

setbe, 41

setc, 41

sete, 41

setg, 41

setge, 41

setl, 41

setle, 41

setna, 41

setnae, 41

setnb, 41

setnbe, 41

setnc, 41

setne, 41

setng, 41

setnge, 41

setnl, 41

setnle, 41

setno, 41

setnp, 42

setns, 42

setnz, 42

seto, 42

setp, 42

setpe, 42

setpo, 42

sets, 42

setz, 42

sfence, 105

sgdt, 119

shift instructions, 40

shl, 40

shld, 40

shlx, 86

shr, 40

shrd, 40

shrx, 86

shufpd, 109

shufps, 103

sidt, 120

SIMD state management instructions, 52

.size, 30

.skip, 30

sldt, 120

.sleb128, 30

smovl, 44

smsw, 120

sqrtpd, 108

sqrtps, 101

sqrtsd, 108

sqrtss, 101

SSE instructions

compare, 101

conversion, 103

data transfer, 99

integer (64-bit SIMD), 104

logical, 102

miscellaneous, 105

MXCSR state management, 104

packed arithmetic, 100

shuffle, 103

unpack, 103

SSE2 instructions

- compare, 109
- conversion, 110
- data movement, 106
- logical, 108
- miscellaneous, 112
- packed arithmetic, 107
- packed single-precision floating-point, 111
- shuffle, 109
- SIMD integer instructions (128-bit), 112
- unpack, 109
- statement, 17
  - empty, 17
- stc, 46
- std, 46
- sti, 46
- stmxcsr, 104
- stos, 45
- stosb, 45
- stosl, 45
- stosw, 45
- str, 120
- string, 20
- .string, 30
- string instructions, 44
- sub, 39
- subpd, 108
- subps, 101
- subsd, 108
- subss, 101
- .symbolic, 30
- sysenter, 120
- sysexit, 120

**T**

- .tbss, 30
- .tcomm, 30
- .tdata, 31
- test, 42
- .text, 31
- .type, 31
- tzcnt, 85

**U**

- U option, 128
- ucomisd, 109
- ucomiss, 102
- ud2, 46
- .uleb128, 31
- unpckhpd, 109
- unpckhps, 103
- unpcklpd, 109
- unpcklps, 103

**V**

- V option, 128
- vaddpd, 54
- vaddps, 54
- vaddsd, 54
- vaddss, 54
- vaddsubpd, 54
- vaddsubps, 54
- vaesdec, 53
- vaesdeclast, 54
- vaesenc, 54
- vaesenclast, 54
- vaesimc, 54
- vaeskeygenassist, 54
- .value, 31
- vandnpd, 54
- vandnps, 54
- vandpd, 55
- vandps, 55
- vblendpd, 55
- vblendps, 55
- vblendvpd, 55
- vblendvps, 55
- vbroadcast(f128|sd|ss), 76
- vbroadcast(sd|ss), 83
- vbroadcasti128, 83
- vcmpcq\_ospd, 55
- vcmpcq\_osps, 57
- vcmpcq\_ossd, 59

---

vcmpeq\_oss, 62  
vcmpeq\_uqpd, 55  
vcmpeq\_uqps, 57  
vcmpeq\_uqsd, 59  
vcmpeq\_uqss, 62  
vcmpeq\_uspd, 55  
vcmpeq\_usps, 57  
vcmpeq\_ussd, 59  
vcmpeq\_uss, 62  
vcmpeqpd, 55  
vcmpeqps, 57  
vcmpeqsd, 60  
vcmpeqss, 62  
vcmpfalse\_ospd, 55  
vcmpfalse\_osp, 57  
vcmpfalse\_ossd, 60  
vcmpfalse\_oss, 62  
vcmpfalsepd, 55  
vcmpfalseps, 57  
vcmpfalsesd, 60  
vcmpfalsess, 62  
vcmpge\_oqpd, 55  
vcmpge\_oqps, 58  
vcmpge\_oqsd, 60  
vcmpge\_oqss, 62  
vcmpgepd, 55  
vcmpgeps, 58  
vcmpgesd, 60  
vcmpgess, 62  
vcmpgt\_oqpd, 55  
vcmpgt\_oqps, 58  
vcmpgt\_oqsd, 60  
vcmpgt\_oqss, 62  
vcmpgtpd, 56  
vcmpgtps, 58  
vcmpgtsd, 60  
vcmpgtss, 62  
vcmpole\_oqpd, 56  
vcmpole\_oqps, 58  
vcmpole\_oqsd, 60  
vcmpole\_oqss, 62  
vcmpolepd, 56  
vcmpoleps, 58  
vcmpolesd, 60  
vcmpoles, 62  
vcmplt\_oqpd, 56  
vcmplt\_oqps, 58  
vcmplt\_oqsd, 60  
vcmplt\_oqss, 62  
vcmpltpd, 56  
vcmpltps, 58  
vcmpltsd, 60  
vcmpltss, 62  
vcmpneq\_oqpd, 56  
vcmpneq\_oqps, 58  
vcmpneq\_oqsd, 60  
vcmpneq\_oqss, 62  
vcmpneq\_ospd, 56  
vcmpneq\_osp, 58  
vcmpneq\_ossd, 60  
vcmpneq\_oss, 63  
vcmpneq\_uspd, 56  
vcmpneq\_usps, 58  
vcmpneq\_ussd, 60  
vcmpneq\_uss, 63  
vcmpneqpd, 56  
vcmpneqps, 58  
vcmpneqsd, 60  
vcmpneqss, 63  
vcmpnge\_uqpd, 56  
vcmpnge\_uqps, 58  
vcmpnge\_uqsd, 61  
vcmpnge\_uqss, 63  
vcmpngepd, 56  
vcmpngeps, 58  
vcmpngesd, 61  
vcmpngess, 63  
vcmpngt\_uqpd, 56  
vcmpngt\_uqps, 58  
vcmpngt\_uqsd, 61  
vcmpngt\_uqss, 63  
vcmpngtpd, 56





---

vmadd213pd, 86  
vmadd213ps, 87  
vmadd213sd, 87  
vmadd213ss, 87  
vmadd231pd, 86  
vmadd231ps, 87  
vmadd231sd, 87  
vmadd231ss, 87  
vmaddsub132pd, 87  
vmaddsub132ps, 88  
vmaddsub213pd, 88  
vmaddsub213ps, 88  
vmaddsub231pd, 88  
vmaddsub231ps, 88  
vfmsub132pd, 88  
vfmsub132ps, 88  
vfmsub132sd, 89  
vfmsub132ss, 89  
vfmsub213pd, 88  
vfmsub213ps, 88  
vfmsub213sd, 89  
vfmsub213ss, 89  
vfmsub231pd, 88  
vfmsub231ps, 89  
vfmsub231sd, 89  
vfmsub231ss, 89  
vfmsubadd132pd, 89  
vfmsubadd132ps, 90  
vfmsubadd213pd, 89  
vfmsubadd213ps, 90  
vfmsubadd231pd, 89  
vfmsubadd231ps, 90  
vfmadd132pd, 90  
vfmadd132ps, 90  
vfmadd132sd, 90  
vfmadd132ss, 91  
vfmadd213pd, 90  
vfmadd213ps, 90  
vfmadd213sd, 91  
vfmadd213ss, 91  
vfmadd231pd, 90  
vfmadd231ps, 90  
vfmadd231sd, 91  
vfmadd231ss, 91  
vfnmsub132pd, 91  
vfnmsub132ps, 91  
vfnmsub132sd, 92  
vfnmsub132ss, 92  
vfnmsub213pd, 91  
vfnmsub213ps, 91  
vfnmsub213sd, 92  
vfnmsub213ss, 92  
vfnmsub231pd, 91  
vfnmsub231ps, 92  
vfnmsub231sd, 92  
vfnmsub231ss, 92  
vgatherdp(d|s), 83  
vgatherqp(d|s), 83  
vhaddpd, 66  
vhaddps, 66  
vhsubpd, 66  
vhsubps, 66  
vinsertf128, 76  
vinserti128, 83  
vinsertps, 66  
vladdq, 66  
vldmxcsr, 66  
vmaskmov(pd|ps), 77  
vmaskmovdqu, 66  
vmaxpd, 66  
vmaxps, 66  
vmaxsd, 66  
vmaxss, 67  
vmcall, 120  
vmclear, 120  
vmfunc, 120  
vminpd, 67  
vminps, 67  
vminsd, 67  
vminss, 67  
vmlaunch, 120  
vmov(q|d), 67

vmovapd, 67  
vmovaps, 67  
vmovddup, 67  
vmovdqa, 67  
vmovdqu, 67  
vmovhlps, 67  
vmovhpd, 67  
vmovhps, 67  
vmovlhps, 68  
vmovlpd, 68  
vmovlps, 68  
vmovmskpd, 68  
vmovmskps, 68  
vmovntdq, 68  
vmovntdqa, 68, 77  
vmovntpd, 68  
vmovntps, 68  
vmovq, 68  
vmovsd, 68  
vmovshdup, 68  
vmovsldup, 68  
vmovss, 68  
vmovupd, 69  
vmovups, 69  
vmvpsadbw, 69, 78  
vmptrl, 121  
vmptrst, 121  
vmread, 121  
vmresume, 120  
vmulpd, 69  
vmulps, 69  
vmulsd, 69  
vmulss, 69  
vmwrite, 121  
vmxoff, 121  
vmxon, 121  
vorpd, 69  
vorps, 69  
vpabs(w|b|d), 69, 78  
vpackss(dw|wb), 69, 78  
vpackusdw, 69, 78  
vpackuswb, 69, 78  
vpadd(q|w|b|d), 69, 78  
vpadds(w|b), 70, 78  
vpaddus(w|b), 70, 78  
vpalignr, 70, 78  
vpand, 70, 78  
vpandn, 70, 78  
vpavg(w|b), 70, 79  
vpblendd, 83  
vpblendvb, 70, 79  
vpblendw, 70, 79  
vpbroadcast(q|w|b|d), 83  
vpcmlulqdq, 77  
vpcmpeq(q|w|b|d), 70, 79  
vpcmpestri, 70  
vpcmpestrm, 70  
vpcmpgt(q|w|b|d), 70, 79  
vpcmpistri, 71  
vpcmpistrm, 71  
vperm2f128, 77  
vperm2i128, 83  
vpermd, 83  
vpermilpd, 77  
vpermilps, 77  
vpermpd, 83  
vpermps, 83  
vpermq, 84  
vpextr(q|b|d), 71  
vpextrw, 71  
vpgatherdd, 84  
vpgatherdq, 84  
vpgatherqd, 84  
vpgatherqq, 84  
vphadd(w|d), 71, 79  
vphaddsw, 71, 79  
vphminposuw, 71  
vphsub(w|d), 71, 79  
vphsubsw, 71, 79  
vpinsr(q|b|w|d), 71  
vpinsrw, 71  
vpmaddubsw, 71, 79

vpmaddwd, 71, 79  
vpmaskmov(q|d), 84  
vpmaws(w|b|d), 71, 79  
vpmawub, 72, 80  
vpmawud, 72, 80  
vpmawuw, 72, 80  
vpmawub, 72, 80  
vpmawud, 72, 80  
vpmawuw, 72, 80  
vpmawmskb, 72, 80  
vpmawsx(bd|bq|bw|dq|wd|wq), 72, 80  
vpmawzx(bd|bq|bw|dq|wd|wq), 72, 80  
vpmuldq, 72, 80  
vpmulhrsw, 72, 80  
vpmulhuw, 73, 80  
vpmulhw, 73, 81  
vpmulld, 73, 81  
vpmullw, 73, 81  
vpmuludq, 73, 81  
vpor, 73, 81  
vpsadbw, 73, 81  
vpshufb, 73, 81  
vpshufd, 73, 81  
vpshufhw, 73, 81  
vpshuflw, 73, 81  
vpsign(w|b|d), 73, 81  
vpsll(q|w|d), 73, 81  
vpslldq, 73, 81  
vpsllv(q|d), 84  
vpsra(w|d), 74, 82  
vpsravd, 84  
vpsrl(q|w|d), 74, 82  
vpsrldq, 74, 82  
vpsrlv(q|d), 84  
vpsub(q|w|b|d), 74, 82  
vpsubs(w|b), 74, 82  
vpsubus(w|b), 74, 82  
vptest, 74  
vpunpckh(bw|dq|qdq|wd), 74, 82  
vpunpckl(bw|dq|qdq|wd), 74, 82  
vpxor, 74, 82  
vrcpps, 74  
vrcpss, 75  
vroundpd, 75  
vroundps, 75  
vroundsd, 75  
vroundss, 75  
vrsqrtps, 75  
vrsqrtss, 75  
vshufpd, 75  
vshufps, 75  
vsqrtpd, 75  
vsqrtps, 75  
vsqrtsd, 75  
vsqrtss, 75  
vstmcsr, 75  
vsubpd, 76  
vsubps, 76  
vsubsd, 76  
vsubss, 76  
vtestp(d|s), 77  
vucomisd, 76  
vucomiss, 76  
vunpckhpd, 76  
vunpckhps, 76  
vunpcklpd, 76  
vunpcklps, 76  
vxorpd, 77  
vxorps, 77  
vzeroall, 77  
vzeroupper, 77

**W**  
wait, 52  
wbinvd, 120  
.weak, 31  
whitespace, 17  
wrfsbase(|l|q), 93

wrgsbase(|l|q), 93  
wrmsr, 120

## **X**

xabort, 118  
xadd, 38  
xbegin(|l|w), 118  
xchg, 38  
xchgA, 38  
-xchip option, 128  
xend, 118  
xlat, 46  
xlatb, 46  
-xmodel option, 129  
xor, 40  
xorpd, 108  
xorps, 102  
xsaveopt(64|), 121  
xtest, 118, 119

## **Y**

-Y{d|m} option, 129  
-YI option, 129

## **Z**

.zero, 31