

**Power Instruction Set Architecture
Preliminary Decimal Floating-Point Architecture**

Note:

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 3.

July, 2007

The facilities discussed in this publication is available on certain System p Processor Complexes. The information published herein should not be construed as implying any intention by IBM to provide these facilities on models other than those described herein.

This publication is provided for use in conjunction with other relevant IBM publications, and IBM makes no warranty, express or implied, relative to its completeness or accuracy. The information in this publication is current as of its publication date but is subject to change without notice.

© Copyright International Business Machines Corporation 1990-2007. All rights reserved.

US Government Users Restricted Rights — Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Preliminary - Subject to change

Notices

IBM Corporation

New Orchard Rd.

Armonk, NY 10504

U.S.A.

Produced in the United States of America

July, 2007

All Rights Reserved

The information in this document is intended to provide guidance for those implementing decimal-floating-point architecture. It discusses findings based on a solution that was created and tested under laboratory conditions. These findings may not be realized in all customer environments, and implementation in such environments may require additional steps, configurations, and performance analysis. The information herein is provided "AS IS" with no warranties, express or implied. This information does not constitute a specification or form part of the warranty for any IBM product. Implementation and certification of the solution rests on the implementation team. The users of this document should always check the latest release information for the applicable product and check the product Web pages for the latest updates and findings.

References in this publication to IBM products or services do not imply that IBM intends to make them available in every country in which IBM operates. Consult your local IBM business contact for information on the products, features and services available in your area.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY, 10504-1785 USA.

Photographs shown are of engineering prototypes.

Changes may be incorporated in production models.

This equipment is subject to all applicable FCC rules and will comply with them upon delivery.

Information concerning non-IBM products was obtained from the suppliers of those products. Questions concerning those products should be directed to suppliers.

IBM hardware products are manufactured from new parts, or new and used parts. Regardless, our warranty terms apply.

Trademarks

IBM, the IBM logo, and Power ISA Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

IEEE is a trademark of the Institute of Electrical and Electronics Engineers, Inc. in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems Inc. in the United States and other Countries.

Other trademarks and registered trademarks are the properties of their respective companies.

Chapter 1. Decimal Floating-Point [Category: Decimal Floating-Point]

1.1 Decimal Floating-Point (DFP) Processor Overview	5	1.5.8.2 Data-Type Conversion	15
1.2 DFP Register Handling	6	1.5.9 Format Operations	15
1.2.1 DFP Usage of Floating-Point Registers	6	1.5.10 DFP Exceptions	16
1.3 DFP Support for Non-DFP Data Types	8	1.5.10.1 Invalid Operation Exception	18
1.4 DFP Number Representation	9	1.5.10.2 Zero Divide Exception	18
1.4.1 DFP Data Format	9	1.5.10.3 Overflow Exception	19
1.4.1.1 Fields Within the Data Format	10	1.5.10.4 Underflow Exception	19
1.4.1.2 Summary of DFP Data Formats	11	1.5.10.5 Inexact Exception	20
1.4.1.3 Preferred DPD Encoding	11	1.5.11 Summary of Normal Rounding And Range Actions	21
1.4.2 Classes of DFP Data	11	1.6 DFP Instruction Descriptions	23
1.5 DFP Execution Model	12	1.6.1 DFP Arithmetic Instructions	24
1.5.1 Rounding	12	1.6.2 DFP Compare Instructions	28
1.5.2 Rounding Mode Specification	13	1.6.3 DFP Test Instructions	31
1.5.3 Formation of Final Result	13	1.6.4 DFP Quantum Adjustment Instructions	34
1.5.3.1 Use of Ideal Exponent	13	1.6.5 DFP Conversion Instructions	41
1.5.4 Arithmetic Operations	14	1.6.5.1 DFP Data-Format Conversion Instructions	41
1.5.4.1 Sign of Arithmetic Result	14	1.6.5.2 DFP Data-Type Conversion Instructions	44
1.5.5 Compare Operations	14	1.6.6 DFP Format Instructions	46
1.5.6 Test Operations	14	1.6.7 DFP Instruction Summary	50
1.5.7 Quantum Adjustment Operations	15		
1.5.8 Conversion Operation	15		
1.5.8.1 Data-Format Conversion	15		

1.1 Decimal Floating-Point (DFP) Processor Overview

This chapter describes the behavior of the decimal floating-point processor, the supported data types, formats, and classes, and the usage of registers. Also included are the execution model, exceptions, and instructions supported by the decimal floating-point processor.

The decimal floating-point (DFP) processor shares the 32 floating-point registers (FPRs) and the floating-point status and control register (FPSCR) with the binary floating-point (BFP) processor. However, the interpretation of data formats in the FPRs, and the meaning of some control and status bits in the FPSCR are different between the BFP and DFP processors.

The DFP processor also shares the condition register (CR) with the fixed-point processor and the BFP processor.

The DFP processor supports three DFP data formats: DFP32 (single precision), DFP64 (double precision), and DFP128 (quad precision). Most operations are performed on the DFP64 or DFP128 format directly. Support for DFP32 is limited to conversion to and from DFP64. For some operations, the DFP processor also supports operands in other data types, including signed or unsigned binary fixed-point data, and signed or unsigned decimal data.

DFP instructions are provided to perform arithmetic, compare, test, quantum-adjustment, conversion, and format operations on operands held in FPRs or FPR pairs.

■ Arithmetic instructions

These instructions perform addition, subtraction, multiplication, and division operations.

■ Compare instructions

These instructions perform a comparison operation on the numerical value of two DFP operands.

■ Test instructions

These instructions test the data class, the data group, the exponent, or the number of significant digits of a DFP operand.

■ Quantum-adjustment instructions

These instructions convert a DFP number to a result in the form that has the designated exponent, which may be explicitly or implicitly specified.

■ Conversion instructions

These instructions perform conversion between different data formats or data types.

■ Format instructions

These instructions facilitate composing or decomposing a DFP operand.

These instructions are described in Section 1.6 “DFP Instruction Descriptions” on page 23.

The three DFP data formats allow finite numbers to be represented with different precision and ranges. Special codes are also provided to represent $+\infty$, $-\infty$, Quiet NaN (Not-a-Number), and Signaling NaN. Operations involving infinities produce results obeying traditional mathematical conventions. NaNs have no mathematical interpretation. The encoding of NaNs allows a diagnostic information field. This diagnostic field may be used to indicate such things as the source of uninitialized variables or the reason an invalid result was produced.

The DFP processor recognizes a set of DFP exceptions which are indicated via bits set in the FPSCR. Additionally, the DFP exception actions depend on the setting of the various exception enable bits in the FPSCR.

The following DFP exceptions are detected by the DFP processor. The reporting exception status bits in the FPSCR are indicated in parentheses.

- Invalid Operation Exception (VX)
 - SNaN (VXSNaN)
 - $\infty - \infty$ (VXISI)
 - $\infty \div \infty$ (VXIDI)
 - $0 \div 0$ (VXZDZ)
 - $\infty \times 0$ (VXIMZ)
 - Invalid Compare (VXVC)
 - Invalid conversion (VXCVI)
- Zero Divide Exception (ZX)
- Overflow Exception (OX)
- Underflow Exception (UX)
- Inexact Exception (XX)

Each DFP exception, and each category of Invalid Operation Exception, has an exception status bit in the FPSCR. In addition, each of the five DFP exceptions has a corresponding enable bit in the FPSCR. These enable bits enable or disable the invocation of the system floating-point enabled exception error handler, and

may affect the setting of some exception status bits in the FPSCR.

The usage of these bits by the DFP processor differs from the usage by the BFP processor. Section 1.5.10 “DFP Exceptions” on page 16 provides a detailed discussion of DFP exceptions, including the effects of the enable bits.

1.2 DFP Register Handling

The following sections describe first how the floating-point registers are utilized by the DFP processor. The subsequent section covers the DFP usage of CR and FPSCR.

1.2.1 DFP Usage of Floating-Point Registers

The DFP processor shares the same 32 64-bit FPRs with the BFP processor. Like the BFP instructions, DFP instructions also use 5-bit fields for designating the FPRs to hold the source or target operands.

When data of the DFP32 format is held in a FPR, it occupies the rightmost 32 bits of the FPR. The *Load Floating-Point as Integer Word Algebraic* instruction is provided to load the rightmost 32 bits of a FPR with a single-word data from storage. The *Store Floating-Point as Integer Word* instruction is available to store the rightmost 32 bits of a FPR to a storage location.

Data of the DFP64 format, 64-bit binary fixed-point values, or 64-bit BCD values is held in a FPR using all 64 bits. Data of 64 bits may be loaded from storage via any of the *Load Floating-Point Double* instructions and stored via any of the *Store Floating-Point Double* instructions.

Data of the DFP128 format or 128-bit BCD values is held in an even-odd FPR pair using all 128 bits. Data of 128 bits must be loaded into the desired even-odd pair of floating-point registers using an appropriate sequence of the *Load Floating-Point Double* instructions and stored using an appropriate sequence of the *Store Floating-Point Double* instructions.

When an even-odd FPR pair is used to hold a 128-bit operand, the even-numbered FPR is used to hold the leftmost doubleword of the operand and the next higher-numbered FPR is used to hold the rightmost doubleword. A DFP instruction designating an odd-numbered FPR for a 128-bit operand is an invalid instruction form.

Programming Note

The *Floating-Point Move* instructions can be used to move operands between FPRs.

The bit definitions for the FPSCR are as follows.

Bit(s)	Description	
0:28	Reserved	
29:31	DFP Rounding Control (DRN) See Section 1.5.2, "Rounding Mode Specification" on page 13.	
	000 Round to Nearest, Ties to Even	
	001 Round toward Zero	39
	010 Round toward +Infinity	
	011 Round toward -Infinity	
	100 Round to Nearest, Ties away from 0	
	101 Round to Nearest, Ties toward 0	
	110 Round to away from Zero	40
	111 Round to Prepare for Shorter Precision	
	<p style="text-align: center;">Programming Note</p> <p>FPSCR₂₈ is reserved for extension of the DRN field, therefore DRN may be set using the <i>mtfsfi</i> instruction to set the rounding mode.</p>	
32	Floating-Point Exception Summary (FX) Every floating-point instruction, except <i>mtfsfi</i> and <i>mtfsf</i> , implicitly sets FPSCR _{FX} to 1 if that instruction causes any of the floating-point exception bits in the FPSCR to change from 0 to 1. <i>mcrfs</i> , <i>mtfsfi</i> , <i>mtfsf</i> , <i>mtfsb0</i> , and <i>mtfsb1</i> can alter FPSCR _{FX} explicitly.	
33	Floating-Point Enabled Exception Summary (FEX) This bit is the OR of all the floating-point exception bits masked by their respective enable bits. <i>mcrfs</i> , <i>mtfsfi</i> , <i>mtfsf</i> , <i>mtfsb0</i> , and <i>mtfsb1</i> cannot alter FPSCR _{FEX} explicitly.	
34	Floating-Point Invalid Operation Exception Summary (VX) This bit is the OR of all the Invalid Operation exception bits. <i>mcrfs</i> , <i>mtfsfi</i> , <i>mtfsf</i> , <i>mtfsb0</i> , and <i>mtfsb1</i> cannot alter FPSCR _{VX} explicitly.	
35	Floating-Point Overflow Exception (OX) See Section 1.5.10.3, "Overflow Exception" on page 19.	
36	Floating-Point Underflow Exception (UX) See Section 1.5.10.4, "Underflow Exception" on page 19.	
37	Floating-Point Zero Divide Exception (ZX) See Section 1.5.10.2, "Zero Divide Exception" on page 18.	
38	Floating-Point Inexact Exception (XX) See Section 1.5.10.5, "Inexact Exception" on page 20. FPSCR _{XX} is a sticky version of FPSCR _{FI} (see below). Thus the following rules completely describe how FPSCR _{XX} is set by a given instruction.	
		<ul style="list-style-type: none"> ■ If the instruction affects FPSCR_{FI}, the new value of FPSCR_{XX} is obtained by ORing the old value of FPSCR_{XX} with the new value of FPSCR_{FI}. ■ If the instruction does not affect FPSCR_{FI}, the value of FPSCR_{XX} is unchanged.
		39 Floating-Point Invalid Operation Exception (SNaN) (VXSNAN) See Section 1.5.10.1, "Invalid Operation Exception" on page 18.
		40 Floating-Point Invalid Operation Exception ($\infty - \infty$) (VXISI) See Section 1.5.10.1.
		41 Floating-Point Invalid Operation Exception ($\infty \div \infty$) (VXIDI) See Section 1.5.10.1.
		442 Floating-Point Invalid Operation Exception ($0 \div 0$) (VXZDZ) See Section 1.5.10.1.
		43 Floating-Point Invalid Operation Exception ($\infty \times 0$) (VXIMZ) See Section 1.5.10.1.
		44 Floating-Point Invalid Operation Exception (Invalid Compare) (VXVC) See Section 1.5.10.1.
		45 Floating-Point Fraction Rounded (FR) The last <i>Arithmetic</i> or <i>Rounding and Conversion</i> instruction incremented the fraction during rounding. See Section 1.5.1, "Rounding" on page 12. This bit is not sticky.
		46 Floating-Point Fraction Inexact (FI) The last <i>Arithmetic</i> or <i>Rounding and Conversion</i> instruction either produced an inexact result during rounding or caused a disabled Overflow Exception. See Section 1.5.1. This bit is not sticky.
		See the definition of FPSCR _{XX} , above, regarding the relationship between FPSCR _{FI} and FPSCR _{XX} .
		47:51 Floating-Point Result Flags (FPRF) This field is set as described below. For arithmetic, rounding, and conversion instructions, the field is set based on the result placed into the target register, except that if any portion of the result is undefined then the value placed into FPRF is undefined.
		47 Floating-Point Result Class Descriptor (C) Arithmetic, rounding, and conversion instructions may set this bit with the FPCC bits, to indicate the class of the result as shown in Figure 1 on page 8.
		48:51 Floating-Point Condition Code (FPCC) Floating-point <i>Compare</i> and <i>DFP Test</i> instruc-

tions set one of the FPCC bits to 1 and the other three FPCC bits to 0. Arithmetic, rounding, and conversion instructions may set the FPCC bits with the C bit, to indicate the class of the result as shown in Figure 1 on page 8. Note that in this case the high-order three bits of the FPCC retain their relational significance indicating that the value is less than, greater than, or equal to zero.

- 48 **Floating-Point Less Than or Negative** (FL or <)
- 49 **Floating-Point Greater Than or Positive** (FG or >)
- 50 **Floating-Point Equal or Zero** (FE or =)
- 51 **Floating-Point Unordered or NaN** (FU or ?)
- 52 Reserved
- 53 **Floating-Point Invalid Operation Exception (Software Request)** (VXSOFT)
This bit can be altered only by *mcrfs*, *mtfsfi*, *mtfsf*, *mtfsb0*, or *mtfsb1*. See Section 1.5.10.1, "Invalid Operation Exception" on page 18.
- 54 Neither used nor changed by DFP.

Programming Note

Although the architecture does not provide a DFP square root instruction, if software simulates such an instruction, it should set bit 54 whenever the source operand of the square root function is invalid.

- 55 **Floating-Point Invalid Operation Exception (Invalid Conversion)** (VXCVI)
See Section 1.5.10.1.
- 56 **Floating-Point Invalid Operation Exception Enable** (VE)
See Section 1.5.10.1.
- 57 **Floating-Point Overflow Exception Enable** (OE)
See Section 1.5.10.3, "Overflow Exception" on page 19.
- 58 **Floating-Point Underflow Exception Enable** (UE)
See Section 1.5.10.4, "Underflow Exception" on page 19.
- 59 **Floating-Point Zero Divide Exception Enable** (ZE)
See Section 1.5.10.2, "Zero Divide Exception" on page 18.
- 60 **Floating-Point Inexact Exception Enable** (XE)
See Section 1.5.10.5, "Inexact Exception" on page 20

- 61 Reserved (not used by DFP)
- 62:63 **Binary Floating-Point Rounding Control** (RN)
See Section 1.5.1, "Rounding" on page 12.
 - 00 Round to Nearest
 - 01 Round toward Zero
 - 10 Round toward +Infinity
 - 11 Round toward -Infinity

Result Flags	Result Value Class
C < > = ?	
0 0 0 0 1	Signaling NaN (DFP only)
1 0 0 0 1	Quiet NaN
0 1 0 0 1	- Infinity
0 1 0 0 0	- Normal Number
1 1 0 0 0	- Subnormal Number
1 0 0 1 0	- Zero
0 0 0 1 0	+ Zero
1 0 1 0 0	+ Subnormal Number
0 0 1 0 0	+ Normal Number
0 0 1 0 1	+ Infinity

Figure 1. Floating-Point Result Flags

1.3 DFP Support for Non-DFP Data Types

In addition to the DFP data types, the DFP processor provides limited support for the following non-DFP data types: signed or unsigned binary fixed-point data, and signed or unsigned decimal data.

In unsigned binary fixed-point data, all bits are used to express the absolute value of the number. For signed binary fixed-point data, the leftmost bit represents the sign, which is followed by the numeric field. Positive numbers are represented in true binary notation with the sign bit set to zero. When the value is zero, all bits are zeros, including the sign bit. Negative numbers are represented in two's complement binary notation with a one in the sign-bit position.

For decimal data, each byte contains a pair of four-bit nibbles; each four-bit nibble contain a binary-coded-decimal (BCD) code. There are two kinds of BCD codes: digit code and sign code. For unsigned decimal data, all nibbles contain a digit code (D) as shown in Figure 2

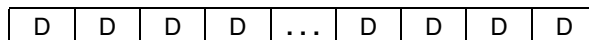


Figure 2. Format for Unsigned Decimal Data

For signed decimal data, the rightmost nibble contains a sign code (S) and all other nibbles contain a digit code as shown in Figure 3.

D	D	D	D	...	D	D	D	S
---	---	---	---	-----	---	---	---	---

Figure 3. Format for Signed Decimal Data

The decimal digits 0-9 have the binary encoding 0000-1001. The preferred plus-sign codes are 1100 and 1111. The preferred minus sign code is 1101. These are the sign codes generated for the results of the *Decode DPD To BCD* instruction. A selection is provided by this instruction to specify which of the two preferred plus sign codes is to be generated. Alternate sign codes are also recognized as valid in the sign position: 1010 and 1110 are alternate sign codes for plus, and 1011 is an alternate sign code for minus. Alternate sign codes are accepted for any source operand, but are not generated as a result by the instruction. When an invalid digit or sign code is detected by the *Encode BCD To DPD* instruction, an invalid-operation exception occurs. A summary of digit and sign codes are provided in Figure 4.

Binary Code	Recognized As	
	Digit	Sign
0000	0	Invalid
0001	1	Invalid
0010	2	Invalid
0011	3	Invalid
0100	4	Invalid
0101	5	Invalid
0110	6	Invalid
0111	7	Invalid
1000	8	Invalid
1001	9	Invalid
1010	Invalid	Plus
1011	Invalid	Minus
1100	Invalid	Plus (preferred; option 1)
1101	Invalid	Minus (preferred)
1110	Invalid	Plus
1111	Invalid	Plus (preferred; option 2)

Figure 4. Summary of BCD Digit and Sign Codes

1.4 DFP Number Representation

A DFP finite number consists of three components: a sign bit (s), a signed exponent (X), and a coefficient (C). The signed exponent is a signed binary integer. The *coefficient* consists of a number of decimal digits, which are to the left of the implied decimal point. The rightmost digit of the coefficient is called the *units* digit. The numerical value of a DFP finite number is repre-

sented as $(-1)^{sign} \times coefficient \times 10^{exponent}$ and the unit value of this number is (1×10^{exponent}) , which is called the *quantum*.

DFP finite numbers are not normalized. This allows leading zeros and trailing zeros to exist in the coefficient. This unnormalized DFP number representation allows some values to have redundant forms; each form represents the DFP number with a different combination of the coefficient value and the exponent value. For example, 1000000×10^5 and 10×10^{10} are two different forms of the same numerical value. A *form* of this number representation carries information about both the numerical value and the quantum of a DFP finite number.

The *significant digits* of a DFP finite number are the digits in the coefficient beginning with the leftmost nonzero digit and ending with the units digit.

1.4.1 DFP Data Format

DFP numbers and NaNs may be represented in FPRs in any of the three data formats: DFP32, DFP64, or DFP128. The contents of each data format represent encoded information. Special codes are assigned to NaNs and infinities. Different formats support different sizes in both coefficient and exponent. Arithmetic, compare, test, quantum-adjustment, and format instructions are provided for the DFP 64 and DFP128 formats only.

The *sign* is encoded as a one bit binary value. *Coefficient* is encoded as an unsigned decimal integer in two distinct parts. The leftmost digit (LMD) of the *coefficient* is encoded as part of the *combination field*; the remaining digits of the *coefficient* are encoded in the *coefficient continuation* field. Similarly, the *exponent* is represented in two parts. However, prior to encoding, the *exponent* is converted to an unsigned binary value called the *biased exponent* by adding a *bias* value which is a constant for each format. The two leftmost bits of the *biased exponent* are encoded in the *combination field* and the remaining 6, 8, or 12 bits (depending on the format) are encoded in the *biased exponent continuation* field.

1.4.1.1 Fields Within the Data Format

The DFP data representation comprises four fields, as diagrammed below for each of the three formats::

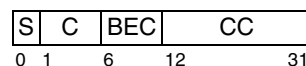


Figure 5. DFP32 format

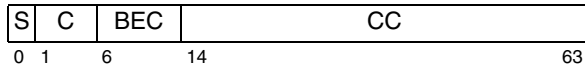


Figure 6. DFP64 format

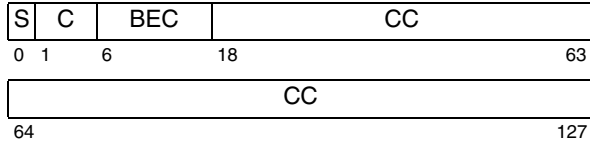


Figure 7. DFP128 format

The fields are defined as follows:

Sign bit (S)

The sign bit is in bit 0 of each format, and is zero for plus and one for minus.

Combination field (C)

This is a 5-bit field which contains the encoding of NaN or infinity, or the two leftmost bits of the biased exponent and the leftmost digit (LMD) of the coefficient. The following tables shows the encoding:

C field	Description
11111	NaN
11110	Infinity
All others	Finite Number (see Figure 9)

Figure 8. Encoding of the C field for Special Symbols

LMD	Leftmost 2-bits of biased exponent		
	00	01	10
0	00000	01000	10000
1	00001	01001	10001
2	00010	01010	10010
3	00011	01011	10011
4	00100	01100	10100
5	00101	01101	10101
6	00110	01110	10110
7	00111	01111	10111
8	11000	11010	11100
9	11001	11011	11101

Figure 9. Encoding of the C field for Finite Numbers

Biased Exponent Continuation field (BEC)

For DFP finite numbers, this field contains the remaining bits of the *biased exponent*. For NaNs, the leftmost bit in this field is used to distinguish a Quiet NaN from a Signaling NaN; the remaining bits in a source operand are ignored and they are set to zeros in a target operand by most operations. For infinities, bits in this field

of a source operand are ignored and they are set to zeros in a target operand by most operations.

Coefficient Continuation field (CC)

For DFP finite numbers, this field contains the remaining *coefficient* digits. For NaNs, this field may be used to contain diagnostic information. For infinities, contents in this field of a source operand are ignored and they are set to zeros in a target operand by most operations. The CC field is a multiple of 10-bit blocks. The multiple depends on the format. Each 10-bit block is called a DPD block and represents three decimal digits, using the Densely Packed Decimal (DPD) encoding defined in Appendix A.

1.4.1.2 Summary of DFP Data Formats

The properties of the three DFP formats are summarized in the following table:

	Format		
	DFP32	DFP64	DFP128
Widths (bits):			
Format	32	64	128
Sign	1	1	1
C	5	5	5
BEC	6	8	12
CC	20	50	110
Exponent:			
Max biased	191	767	12,287
Max (X_{max})	90	369	6111
Min (X_{min})	-101	-398	-6176
Bias	101	398	6176
Precision (p) (digits)			
	7	16	34
Magnitude:			
Max normal number, N_{max}	$(10^7 - 1) \times 10^{90}$	$(10^{16} - 1) \times 10^{369}$	$(10^{34} - 1) \times 10^{6111}$
Min normal number, N_{min}	1×10^{-95}	1×10^{-383}	1×10^{-6143}
Min subnormal number, D_{min}	1×10^{-101}	1×10^{-398}	1×10^{-6176}

Figure 10. Summary of DFP Formats

1.4.1.3 Preferred DPD Encoding

Execution of DFP instructions decodes source operands from DFP data formats to an internal format for

processing, and encodes the operation result before the final result is returned as the target operand.

As part of the decoding process, DPD blocks in the CC field of source operands are decoded to their corresponding BCD digit codes using the DPD-to-BCD decoding algorithm. As part of the encoding process, BCD digit codes to be stored into the CC field of the target operand are encoded into DPD blocks using the BCD-to-DPD encoding algorithm. Both the decoding and encoding algorithms are defined in Appendix A.

As explained in Appendix A, there are eight 3-digit decimal values that have redundant DPD codes and one preferred DPD code. All redundant DPD codes are recognized in source operands for the associated 3-digit decimal number. Only the preferred DPD code is generated by DFP operations for the CC field of the target operand.

1.4.2 Classes of DFP Data

There are six classes of DFP data, which include numerical and nonnumeric entities. The numerical entities include zero, subnormal number, normal number, and infinity data classes. The nonnumeric entities include quiet and signaling NaNs data classes. The value of a DFP finite number, including zero, subnormal number, and normal number, is a quantization of the real number based on the data format. The *Test Data Class* instruction may be used to determine the class of a DFP operand. In general, an operation that returns a DFP result sets the $FPSCR_{FPRF}$ field to indicate the data class of the result.

The following tables show the value ranges for finite-number data classes, and the codes for NaNs and infinity.

Data Class	Sign	Magnitude
Zero	±	0*
Subnormal	±	$D_{min} \leq X < N_{min}$
Normal	±	$N_{min} \leq Y \leq N_{max}$
* The coefficient is zero and the exponent is any representable value		

Figure 11. Value Ranges for Finite Number Data Classes

Data Class	Sign	C	BEC	CC
Infinity	±	1111 0	xxx . . . xxx	xxx . . . xxx
Quiet NaN	±	1111 1	0xx . . . xxx	xxx . . . xxx
Signaling NaN	±	1111 1	1xx . . . xxx	xxx . . . xxx
x Don't care				

Figure 12. Encoding of NaNs and Infinity Data Classes

Zeros

Zeros have a zero coefficient and any representable value in the exponent. A +0 is distinct from -0, and zeros with different exponents are distinct, except that comparison treats them as equal.

Subnormal Numbers

Subnormal numbers have values that are smaller than N_{min} and greater than zero in magnitude.

Normal Numbers

A normal number is a nonzero finite number whose magnitude is between N_{min} and N_{max} inclusively.

Infinities

An infinity is represented by 0b11110 in the combination field. When an operation is defined to generate an infinity as the result, a default infinity is sometimes supplied. A default infinity has all bits in the BEC and CC fields set to zeros.

When used as a source operand, the contents of the BEC and the CC fields of an infinity are usually ignored. In all cases, the BEC field of a generated infinity contains all zeros.

Infinities can participate in most arithmetic operations and give a consistent result. In comparisons, any $+\infty$ compares greater than any finite number, and any $-\infty$ compares less than any finite number. All $+\infty$ are compared equal and all $-\infty$ are compared equal.

Signaling and Quiet NaNs

A NaN (Not-a-Number) is represented by the bit pattern 0b11111 in the combination field. There are two types

of NaNs, Signaling and Quiet. A Signaling NaN (SNaN) is distinguished from a Quiet NaN (QNaN) by the leftmost bit in the BEC field: '0' for the QNaN and '1' for the SNaN. A special QNaN is sometimes supplied as the *default QNaN* for a disabled invalid-operation exception; it has a plus sign, and all bits in the BEC field and the CC field are set to zeros.

Normally, source QNaNs are *propagated* during operations so that they will remain visible at the end. When a QNaN is propagated, the sign is preserved, the decimal value of the CC field is preserved but is reencoded using the preferred DPD codes, and the contents in the BEC field are set to zero.

A source SNaN generally causes an invalid-operation exception. If the exception is disabled, the SNaN is converted to a corresponding QNaN which is propagated. The differences between an SNaN and the corresponding QNaN are that the contents of the BEC field of the QNaN are set to zero and the same decimal value of the CC field is reencoded using the preferred DPD codes in the QNaN. For some format-conversion instructions, a source SNaN does not cause an invalid-operation exception, and an SNaN is returned as the target operand.

When more than one source operands are NaNs and a NaN is to be propagated, if all source operands are QNaNs, then the QNaN in FRA is propagated; if all source operands are SNaNs, then the corresponding QNaN of the SNaN in FRA is propagated; if they are different kind of NaNs, then the corresponding QNaN of the SNaN is propagated.

tional restrictions, and the permissible resultant values are a subset of the values representable in the target format.

Rounding sets FPSCR bits FR and FI. When an inexact exception occurs, FI is set to one; otherwise, FI is set to zero. When an inexact exception occurs and if the rounded result is greater in magnitude than the intermediate result, then FR is set to 1; otherwise, FR is set to zero. The exception is the *Round to FP Integer Without Inexact* instruction, which always sets FR and FI to zero. Rounding may cause an overflow exception or underflow exception; it may also cause an inexact exception.

Refer to Figure 13 below for rounding. Let Z be the intermediate result of a DFP operation. Z may or may not fit in the destination's precision. If Z is exactly one of the permissible representable resultant values, then the final result in all rounding modes is Z. Otherwise, either Z1 or Z2 is chosen to approximate the result, where Z1 and Z2 are the next larger and smaller permissible resultant values, respectively.

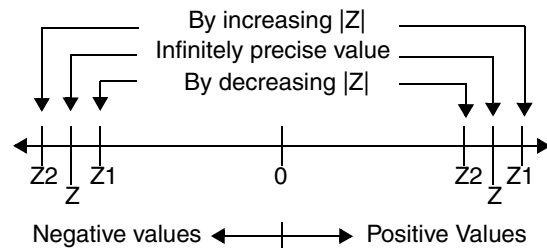


Figure 13. Rounding

1.5 DFP Execution Model

DFP operations are performed as if they first produce an intermediate result correct to infinite precision and with unbounded range. The intermediate result is then rounded to the destination's precision according to one of the eight DFP rounding modes. If the rounded result has only one form, it is delivered as the final result; if the rounded result has redundant forms, then an *ideal exponent* is used to select the form of the final result. The ideal exponent determines the form, not the value, of the final result. (See Section 1.5.3 "Formation of Final Result" on page 13.)

1.5.1 Rounding

Rounding takes a number regarded as infinitely precise and, if necessary, modifies it to fit the destination's precision. The destination's precision of an operation defines the set of permissible resultant values. For most operations, the destination's precision is the target-format precision and the permissible resultant values are those values representable in the target format. For some special operations, the destination precision is constrained by both the target format and some addi-

Round to Nearest, Ties to Even

Choose the value that is closer to Z (Z1 or Z2). In case of a tie, choose the one whose units digit would have been even in the form with the largest common quantum of the two permissible resultant values. However, an infinitely precise result with magnitude at least $(N_{max} + 0.5Q(N_{max}))$ is rounded to infinity with no change in sign; where $Q(N_{max})$ is the quantum of N_{max} .

Round toward 0

Choose the smaller in magnitude (Z1 or Z2).

Round toward $+\infty$

Choose Z1.

Round toward $-\infty$

Choose Z2.

Round to Nearest, Ties away from 0

Choose the value that is closer to Z (Z1 or Z2). In case of a tie, choose the larger in magnitude (Z1 or Z2). However, an infinitely precise result with magnitude at least $(N_{max} + 0.5Q(N_{max}))$ is rounded to infinity with no change in sign; where $Q(N_{max})$ is the quantum of N_{max} .

Round to Nearest, Ties toward 0

Choose the value that is closer to Z (Z1 or Z2). In case of a tie, choose the smaller in magnitude (Z1 or Z2). However, an infinitely precise result with magnitude greater than $(N_{max} + 0.5Q(N_{max}))$ is rounded to infinity with no change in sign; where $Q(N_{max})$ is the quantum of N_{max} .

Round away from 0

Choose the larger in magnitude (Z1 or Z2).

Round to prepare for shorter precision

Choose the smaller in magnitude (Z1 or Z2). If the selected value is inexact and the units digit of the selected value is either 0 or 5, then the digit is incremented by one and the incremented result is delivered. In all other cases, the selected value is delivered. When a value has redundant forms, the units digit is determined by using the form that has the smallest exponent.

1.5.2 Rounding Mode Specification

Unless otherwise specified in the instruction definition, the rounding mode used by an operation is specified in the DFP rounding control (DRN) field of the FPSCR. The eight DFP rounding modes are encoded in the DRN field as specified in the table below.

DRN	Rounding Mode
000	Round to Nearest, Ties to Even
001	Round toward 0
010	Round toward +Infinity
011	Round toward -Infinity
100	Round to Nearest, Ties away from 0
101	Round to Nearest, Ties toward 0
110	Round away from 0
111	Round for Prepare for Shorter Precision

Figure 14. Encoding of DFP Rounding-Mode Control (DRN)

For the quantum-adjustment, a 2-bit immediate field, called RMC (*Rounding Mode Control*), in the instruction specifies the rounding mode used. The RMC field may contain a primary encoding or a secondary encoding. For *Quantize*, *Quantize Immediate*, and *Reround*, the RMC field contains the primary encoding. For *Round to FP Integer* the field contains either encoding, depending on the setting of a RMC-encoding-selection

bit. The following tables define the primary encoding and the secondary encoding.

Primary RMC	Rounding Mode
00	Round to nearest, ties to even
01	Round toward 0
10	Round to nearest, ties away from 0
11	Round according to $FPSCR_{DRN}$

Figure 15. Primary Encoding of Rounding-Mode Control

Secondary RMC	Rounding Mode
00	Round to $+\infty$
01	Round to $-\infty$
10	Round away from 0
11	Round to nearest, ties toward 0

Figure 16. Secondary Encoding of Rounding-Mode Control

1.5.3 Formation of Final Result

An ideal exponent is defined for each DFP instruction that returns a DFP data operand.

1.5.3.1 Use of Ideal Exponent

For all DFP operations,

- if the rounded intermediate result has only one form, then that form is delivered as the final result.
- if the rounded intermediate result has redundant forms and is exact, then the form with the exponent closest to the ideal exponent is delivered.
- if the rounded intermediate result has redundant forms and is inexact, then the form with the smallest exponent is delivered.

The following table specifies the ideal exponent for each instruction.

Operations	Ideal Exponent
Add	$\min(E(FRA), E(FRB))$
Subtract	$\min(E(FRA), E(FRB))$
Multiply	$E(FRA) + E(FRB)$
Divide	$E(FRA) - E(FRB)$
Quantize-Immediate	See Instruction Description
Quantize	$E(FRA)$
Reround	See Instruction Description
Round to FP Integer	$\max(0, E(FRA))$
Convert to DFP64	$E(FRA)$
Convert to DFP128	$E(FRA)$
Round to DFP32	$E(FRA)$
Round to DFP64	$E(FRA)$
Convert from Fixed	0
Encode BCD to DPD	0
Insert Biased Exponent	$E(FRA)$
Notes: E(x) - exponent of the DFP operand in register x.	

Figure 17. Summary of Ideal Exponents

1.5.4 Arithmetic Operations

Four arithmetic operations are provided: Add, Subtract, Multiply, and Divide.

1.5.4.1 Sign of Arithmetic Result

The following rules govern the sign of an arithmetic operation when the operation does not yield an exception. They apply even when the operands or results are zeros or infinities.

- The sign of the result of an add operation is the sign of the source operand having the larger absolute value. If both source operands have the same sign, the sign of the result of an add operation is the same as the sign of the source operands. When the sum of two operands with opposite signs is exactly zero, the sign of the result is positive in all rounding modes except Round toward $-\infty$, in which case the sign is negative.
- The sign of the result of the subtract operation $x - y$ is the same as the sign of the result of the add operation $x + (-y)$.
- The sign of the result of a multiply or divide operation is the exclusive-OR of the signs of the source operands.

1.5.5 Compare Operations

Two sets of instructions are provided for comparing numerical values: *Compare Ordered* and *Compare Unordered*. In the absence of NaNs, these instructions work the same. These instructions work differently when either of the followings is true:

1. At least one source operand of the instruction is an SNaN and the invalid-operation exception is disabled.
2. When there is no SNaN in any source operand, at least one source operand of the instruction is a QNaN

In case 1, *Compare Unordered* recognizes an invalid-operation exception and sets the $FPSCR_{VXSNAN}$ flag, but *Compare Ordered* recognizes the exception and sets both the $FPSCR_{VXSNAN}$ and $FPSCR_{VXVC}$ flags. In case 2, *Compare Unordered* does not recognize an exception, but *Compare Ordered* recognizes an invalid-operation exception and sets the $FPSCR_{VXVC}$ flag.

For finite numbers, comparisons are performed on values, that is, all redundant forms of a DFP number are treated equal.

Comparisons are always exact and cannot cause an inexact exception.

Comparison ignores the sign of zero, that is, $+0$ equals -0 .

Infinities with like sign compare equal, that is, $+\infty$ equals $+\infty$, and $-\infty$ equals $-\infty$.

A NaN compares as unordered with any other operand, whether a finite number, an infinity, or another NaN, including itself.

Execution of a compare instruction always completes, regardless of whether any DFP exception occurs or not, and whether the exception is enabled or not.

1.5.6 Test Operations

Four kinds of test operation are provided: *Test Data Class*, *Test Data Group*, *Test Exponent*, and *Test Significance*.

The *Test Data Class* instruction examines the contents of a source operand and determines if the operand is one of the specified data classes. The test result and the sign of the source operand are indicated in the $FPSCR_{FPCC}$ field and CR field BF.

The *Test Data Group* instruction examines the contents of a source operand and determines if the operand is one of the specified data groups. The test result and the sign of the source operand are indicated in the $FPSCR_{FPCC}$ field and CR field BF.

The *Test Exponent* instruction compares the exponent of the two source operands. The test operation ignores the sign and coefficient of operands. Infinities compare

equal, and NaNs compare equal. The test result is indicated in the $\text{FPSCR}_{\text{FPCC}}$ field and CR field BF.

The *Test Significance* instruction compares the number of significant digits of one source operand with the referenced number of significant digits in another source operand. The test result is indicated in the $\text{FPSCR}_{\text{FPCC}}$ field and CR field BF.

Execution of a test instruction does not cause any DFP exception.

1.5.7 Quantum Adjustment Operations

Four kinds of quantum-adjustment operations are provided: *Quantize*, *Quantize Immediate*, *Reround*, and *Round To FP Integer*. Each of them has an immediate field which specifies whether the rounding mode in FPSCR or a different one is to be used.

The *Quantize* instruction is used to adjust a DFP number to the form that has the specified target exponent. The *Quantize Immediate* instruction is similar to the *Quantize* instruction, except that the target exponent is specified in a 5-bit immediate field as a signed binary integer and has a limited range.

The *Reround* instruction is used to simulate a DFP operation of a precision other than that of DFP64 or DFP128. For the *Reround* instruction to produce a result which accurately reflects that which would have resulted from a DFP operation of the desired precision d in the range {1: 33} inclusively, the following conditions must be met:

- The precision of the preceding DFP operation must be at least one digit larger than d .
- The rounding mode used by the preceding DFP operation must be *round-to-prepare-for-shorter-precision*.

The *Round To FP Integer* instruction is used to round a DFP number to an integer value of the same format. The target exponent is implicitly specified, and is greater than or equal to zero.

1.5.8 Conversion Operation

There are two kinds of conversion operations: data-format conversion and data-type conversion.

1.5.8.1 Data-Format Conversion

The instructions *Convert To DFP64* and *Convert To DFP128* converts DFP operands to wider formats; the instructions *Round To DFP32* and *Round To DFP64* converts DFP operands to narrower formats.

When converting a finite number to a wider format, the result is exact. When converting a finite number to a narrower format, the source operand is rounded to the

target-format precision, which is specified by the instruction, not by the target register size.

When converting a finite number, the ideal exponent of the result is the source exponent.

Conversion of an infinity or NaN to a different format does not preserve the source BEC field. When the result is an infinity or QNaN, the contents of the target BEC field are set to all zeros. When the result is an SNaN, the leftmost bit in the target BEC field is set to one and all other bits are set to zeros.

When converting a NaN to a wider format or when converting an infinity from DFP32 to DFP64, digits in the source CC field are reencoded using the preferred DPD codes with sufficient zeros appended on the left to form the target CC field. When converting a NaN to a narrower format or when converting an infinity from DFP64 to DFP32, the appropriate number of leftmost digits of the source CC field are removed and the remaining digits of the field are reencoded using the preferred DPD codes to form the target CC field.

When converting an infinity between DFP64 and DFP128, a default infinity with the same sign is produced.

When converting an SNaN between DFP32 and DFP64, it is converted to an SNaN without causing an invalid-operation exception. When converting an SNaN between DFP64 and DFP128, the invalid-operation exception occurs; if the invalid-operation exception is disabled, the result is converted to the corresponding QNaN.

1.5.8.2 Data-Type Conversion

The instructions *Convert From Fixed* and *Convert To Fixed* are provided to convert a number between the DFP data type and the signed 64-bit binary-integer data type.

Conversion of a signed 64-bit binary integer to a DFP128 number is always exact.

Conversion of a DFP number to a signed 64-bit binary integer results in an invalid-operation exception when the converted value does not fit into the target format, or when the source operand is an infinity or NaN. When the exception is disabled, the most positive integer is returned if the source operand is a positive number or $+\infty$, and a most negative integer is returned if the source operand is a negative number, $-\infty$, or NaN.

1.5.9 Format Operations

The format instructions are provided to facilitate composing or decomposing a DFP number, and consist of *Encode BCD To DPD*, *Decode DPD To BCD*, *Extract Biased Exponent*, *Insert Biased Exponent*, *Shift Coefficient Left Immediate*, and *Shift Coefficient Right Immediate*. A source operand of SNaN does not cause an

invalid-operation exception, and an SNaN may be produced as the target operand.

1.5.10 DFP Exceptions

This architecture defines the following DFP exceptions:

- Invalid Operation Exception
 - SNaN
 - $\infty - \infty$
 - $\infty \div \infty$
 - $0 \div 0$
 - $\infty \times 0$
 - Invalid Compare
 - Invalid Conversion
- Zero Divide Exception
- Overflow Exception
- Underflow Exception
- Inexact Exception

These exceptions may occur during execution of a DFP instruction.

Each DFP exception, and each category of the Invalid Operation Exception, has an exception status bit in the FPSCR. In addition, each DFP exception has a corresponding enable bit in the FPSCR. The exception status bit indicates occurrence of the corresponding exception. If an exception occurs, the corresponding enable bit governs the result produced by the instruction and, in conjunction with the FE0 and FE1 bits (see the discussion of FE0 and FE1 below), whether and how the system floating-point enabled exception error handler is invoked. (In general, the enabling specified by the enable bit is of invoking the system error handler, not of permitting the exception to occur. The occurrence of an exception depends only on the instruction and its source operands, not on the setting of any control bits. The only deviation from this general rule is that the occurrence of an Underflow Exception may depend on the setting of the enable bit.)

A single instruction, other than *mtfsfi* or *mtfsf*, may set more than one exception bit only in the following cases:

- Inexact Exception may be set with Overflow Exception.
- Inexact Exception may be set with Underflow Exception.
- Invalid Operation Exception (SNaN) may be set with Invalid Operation Exception (Invalid Compare) for *Compare Ordered* instructions
- Invalid Operation Exception (SNaN) may be set with Invalid Operation Exception (Invalid Conversion) for *Convert To Fixed* instructions.

When an exception occurs the instruction execution may be completed or partially completed, depending on the exception and the operation.

For all instructions, except for the Compare and Test instructions, the following exceptions cause the instruction execution to be partially completed. That is, setting

of CR field 1 (when Rc=1) and exception status flags is performed, but no result is stored into the target FPR or FPR pair. For Compare and Test instructions, instruction execution is always completed, regardless of whether any DFP exception occurs or not, and whether the exception is enabled or not.

- Enabled Invalid Operation
- Enabled Zero Divide

For the remaining kinds of exception, instruction execution is completed, a result, if specified by the instruction, is generated and stored into the target FPR or FPR pair, and appropriate status flags are set. The result may be a different value for the enabled and disabled conditions for some of these exceptions. The kinds of exception that deliver a result in target FPR are the following:

- Disabled Invalid Operation
- Disabled Zero Divide
- Disabled Overflow
- Disabled Underflow
- Disabled Inexact
- Enabled Overflow
- Enabled Underflow
- Enabled Inexact

Subsequent sections define each of the DFP exceptions and specify the action that is taken when they are detected.

The IEEE standard specifies the handling of exceptional conditions in terms of “traps” and “trap handlers”. In this architecture, an FPSCR exception enable bit of 1 causes generation of the result value specified in the IEEE standard for the “trap enabled” case: the expectation is that the exception will be detected by software, which will revise the result. An FPSCR exception enable bit of 0 causes generation of the “default result” value specified for the “trap disabled” (or “no trap occurs” or “trap is not implemented”) case: the expectation is that the exception will not be detected by software, which will simply use the default result. The result to be delivered in each case for each exception is described in the sections below.

The IEEE default behavior when an exception occurs is to generate a default value and not to notify software. In this architecture, if the IEEE default behavior when an exception occurs is desired for all exceptions, all FPSCR exception enable bits should be set to 0 and Ignore Exceptions Mode (see below) should be used. In this case the system floating-point enabled exception error handler is not invoked, even if DFP exceptions occur: software can inspect the FPSCR exception bits if necessary, to determine whether exceptions have occurred.

In this architecture, if software is to be notified that a given kind of exception has occurred, the corresponding FPSCR exception enable bit must be set to 1 and a mode other than Ignore Exceptions Mode must be

used. In this case the system floating-point enabled exception error handler is invoked if an enabled DFP exception occurs. The system floating-point enabled exception error handler is also invoked if a *Move To FPSCR* instruction causes an exception bit and the corresponding enable bit both to be 1; the *Move To FPSCR* instruction is considered to cause the enabled exception.

The FE0 and FE1 bits control whether and how the system floating-point enabled exception error handler is invoked if an enabled DFP exception occurs. The location of these bits and the requirements for altering them are described in Book III, *PowerPC AS Operating Environment Architecture*. (The system floating-point enabled exception error handler is never invoked because of a disabled DFP exception.) The effects of the four possible settings of these bits are as follows.

FE0	FE1	Description
0	0	Ignore Exceptions Mode DFP exceptions do not cause the system floating-point enabled exception error handler to be invoked.
0	1	Imprecise Nonrecoverable Mode The system floating-point enabled exception error handler is invoked at some point at or beyond the instruction that caused the enabled exception. It may not be possible to identify the excepting instruction or the data that caused the exception. Results produced by the excepting instruction may have been used by or may have affected subsequent instructions that are executed before the error handler is invoked.
1	0	Imprecise Recoverable Mode The system floating-point enabled exception error handler is invoked at some point at or beyond the instruction that caused the enabled exception. Sufficient information is provided to the error handler that it can identify the excepting instruction and the operands, and correct the result. No results produced by the excepting instruction have been used by or have affected subsequent instructions that are executed before the error handler is invoked.
1	1	Precise Mode The system floating-point enabled exception error handler is invoked precisely at the instruction that caused the enabled exception.

In all cases, the question of whether a DFP result is stored, and what value is stored, is governed by the FPSCR exception enable bits, as described in subsequent sections, and is not affected by the value of the FE0 and FE1 bits.

In all cases in which the system floating-point enabled exception error handler is invoked, all instructions before the instruction at which the system floating-point enabled exception error handler is invoked have completed, and no instruction after the instruction at which the system floating-point enabled exception error handler is invoked has begun execution. (Recall that, for the two Imprecise modes, the instruction at which the system floating-point enabled exception error handler is invoked need not be the instruction that caused the exception.) The instruction at which the system floating-point enabled exception error handler is invoked has not been executed unless it is the excepting instruction, in which case it has been executed if the exception is not among those listed on page 16 as suppressed.

Programming Note

In any of the three non-Precise modes, a *Floating-Point Status and Control Register* instruction can be used to force any exceptions, due to instructions initiated before the *Floating-Point Status and Control Register* instruction, to be recorded in the FPSCR. (This forcing is superfluous for Precise Mode.)

In either of the Imprecise modes, a *Floating-Point Status and Control Register* instruction can be used to force any invocations of the system floating-point enabled exception error handler, due to instructions initiated before the *Floating-Point Status and Control Register* instruction, to occur. (This forcing has no effect in Ignore Exceptions Mode, and is superfluous for Precise Mode.)

In order to obtain the best performance across the widest range of implementations, the programmer should obey the following guidelines.

- If the IEEE default results are acceptable to the application, Ignore Exceptions Mode should be used with all FPSCR exception enable bits set to 0.
- If the IEEE default results are not acceptable to the application, Imprecise Nonrecoverable Mode should be used, or Imprecise Recoverable Mode if recoverability is needed, with FPSCR exception enable bits set to 1 for those exceptions for which the system floating-point enabled exception error handler is to be invoked.
- Ignore Exceptions Mode should not, in general, be used when any FPSCR exception enable bits are set to 1.
- Precise Mode may degrade performance in some implementations, perhaps substantially, and therefore should be used only for debugging and other specialized applications.

1.5.10.1 Invalid Operation Exception

Definition

An Invalid Operation Exception occurs when an operand is invalid for the specified DFP operation. The invalid DFP operations are:

- Any DFP operation on a signaling NaN (SNaN), except for *Test, Round To DFP32, Convert To DFP64, Decode DPD To BCD, Extract Biased Exponent, Insert Biased Exponent, Shift Coefficient Left Immediate, and Shift Coefficient Right Immediate*
- For add or subtract operations, magnitude subtraction of infinities $(+\infty) + (-\infty)$
- Division of infinity by infinity $(\infty \div \infty)$
- Division of zero by zero $(0 \div 0)$
- Multiplication of infinity by zero $(\infty \times 0)$
- Ordered comparison involving a NaN (Invalid Compare)
- The *Quantize* operation detects that the coefficient associated with the specified target exponent would have more significant digits than the target-format precision
- For the *Quantize* operation, when one source operand specifies an infinity and the other specifies a finite number
- The *Reround* operation detects that the target exponent associated with the specified target significance would be greater than X_{\max}
- The *Encode BCD To DPD* operation detects an invalid BCD digit or sign code
- The *Convert To Fixed* operation involving a number too large in magnitude to be represented in the target format, or involving a NaN

Programming Note

In addition, an Invalid Operation Exception occurs if software explicitly requests this by executing an *mtfsfi*, *mtfsf*, or *mtfsb1* instruction that sets $FPSCR_{VXSOFT}$ to 1 (Software Request). The purpose of $FPSCR_{VXSOFT}$ is to allow software to cause an Invalid Operation Exception for a condition that is not necessarily associated with the execution of a DFP instruction. For example, it might be set by a program that computes a square root, if the source operand is negative.

Action

The action to be taken depends on the setting of the Invalid Operation Exception Enable bit of the FPSCR.

When Invalid Operation Exception is enabled ($FPSCR_{VE}=1$) and Invalid Operation occurs, the following actions are taken:

1. One or two Invalid Operation Exceptions are set:

$FPSCR_{VXSNAN}$	(if SNaN)
$FPSCR_{VXISI}$	(if $\infty - \infty$)

- | | |
|-----------------|----------------------------|
| $FPSCR_{VXIDI}$ | (if $\infty \div \infty$) |
| $FPSCR_{VXZDZ}$ | (if $0 \div 0$) |
| $FPSCR_{VXIMZ}$ | (if $\infty \times 0$) |
| $FPSCR_{VXVC}$ | (if invalid comp) |
| $FPSCR_{VXCVI}$ | (if invalid conversion) |

2. If the operation is an arithmetic, quantum-adjustment, conversion, or format, the target FPR is unchanged, $FPSCR_{FR FI}$ are set to zero, and $FPSCR_{FPRF}$ is unchanged.
3. If the operation is a compare, $FPSCR_{FR FIC}$ are unchanged, and $FPSCR_{FPCC}$ is set to reflect unordered.

When Invalid Operation Exception is disabled ($FPSCR_{VE}=0$) and Invalid Operation occurs, the following actions are taken:

1. One or two Invalid Operation Exceptions are set:

$FPSCR_{VXSNAN}$	(if SNaN)
$FPSCR_{VXISI}$	(if $\infty - \infty$)
$FPSCR_{VXIDI}$	(if $\infty \div \infty$)
$FPSCR_{VXZDZ}$	(if $0 \div 0$)
$FPSCR_{VXIMZ}$	(if $\infty \times 0$)
$FPSCR_{VXVC}$	(if invalid comp)
$FPSCR_{VXCVI}$	(if invalid conversion)
2. If the operation is an arithmetic, quantum-adjustment, *Round to DFP64, Convert to DFP128, or format* the target FPR is set to a Quiet NaN, $FPSCR_{FR FI}$ are set to zero, $FPSCR_{FPRF}$ is set to indicate the class of the result (Quiet NaN)
3. If the operation is a *Convert To Fixed* the target FPR is set as follows:
 - FRT is set to the most positive 64-bit binary integer if the operand in FRB is a positive or $+\infty$, and to the most negative 64-bit binary integer if the operand in FRB is a negative number, $-\infty$, or NaN.
 - $FPSCR_{FR FI}$ are set to zero
 - $FPSCR_{FPRF}$ is unchanged
4. If the operation is a compare, $FPSCR_{FR FIC}$ are unchanged, $FPSCR_{FPCC}$ is set to reflect unordered

1.5.10.2 Zero Divide Exception

Definition

A Zero Divide Exception occurs when a Divide instruction is executed with a zero divisor value and a finite nonzero dividend value.

Action

The action to be taken depends on the setting of the Zero Divide Exception Enable bit of the FPSCR.

When Zero Divide Exception is enabled ($FPSCR_{ZE}=1$) and Zero Divide occurs, the following actions are taken:

1. Zero Divide Exception is set
 $FPSCR_{ZX} \leftarrow 1$
2. The target FPR is unchanged
3. $FPSCR_{FR FI}$ are set to zero
4. $FPSCR_{FPRF}$ is unchanged

When Zero Divide Exception is disabled ($FPSCR_{ZE}=0$) and Zero Divide occurs, the following actions are taken:

1. Zero Divide Exception is set
 $FPSCR_{ZX} \leftarrow 1$
2. The target FPR is set to $\pm\infty$, where the sign is determined by the XOR of the signs of the operands
3. $FPSCR_{FR FI}$ are set to zero
4. $FPSCR_{FPRF}$ is set to indicate the class and sign of the result ($\pm\infty$)

1.5.10.3 Overflow Exception

Definition

An overflow exception occurs whenever the target format's largest finite number is exceeded in magnitude by what would have been the rounded result if the exponent range were unbounded.

Action

Except for *Reround*, the following describes the handling of the IEEE overflow exception condition. The *Reround* operation does not recognize an overflow exception condition.

The action to be taken depends on the setting of the Overflow Exception Enable bit of the FPSCR.

When Overflow Exception is enabled ($FPSCR_{OE}=1$) and overflow occurs, the following actions are taken:

1. Overflow Exception is set
 $FPSCR_{OX} \leftarrow 1$
2. The infinitely precise result is divided by 10^α . That is, the exponent adjustment α is subtracted from the exponent. This is called the *wrapped result*. The exponent adjustment for all operations, except for *Round To DFP32* and *Round To DFP64*, is 576 for DFP64 and 9216 for DFP128. For *Round To DFP32* and *Round To DFP64*, the exponent adjustment is 192 for the source format of DFP64 and 3072 for the source format of DFP128.
3. The wrapped result is rounded to the target-format precision. This is called the *wrapped rounded result*.
4. If the wrapped rounded result has only one form, it is the delivered result. If the wrapped rounded result has redundant forms and is exact, the result of the form that has the exponent closest to the wrapped ideal exponent is returned. If the wrapped rounded result has redundant forms and is inexact, the result of the form that has the smallest exponent is returned. The wrapped ideal exponent is

the result of subtracting the exponent adjustment from the ideal exponent.

5. $FPSCR_{FPRF}$ is set to indicate the class and sign of the result (\pm Normal Number)

When Overflow Exception is disabled ($FPSCR_{OE}=0$) and overflow occurs, the following actions are taken:

1. Overflow Exception is set
 $FPSCR_{OX} \leftarrow 1$
2. Inexact Exception is set
 $FPSCR_{XX} \leftarrow 1$
3. The result is determined by the rounding mode and the sign of the intermediate result as follows.

Rounding Mode	Sign of intermediate result	
	Plus	Minus
Round to Nearest, Ties to Even	$+\infty$	$-\infty$
Round toward 0	$+N_{max}$	$-N_{max}$
Round toward $+\infty$	$+\infty$	$-N_{max}$
Round toward $-\infty$	$+N_{max}$	$-\infty$
Round to Nearest, Ties away from 0	$+\infty$	$-\infty$
Round to Nearest, Ties toward 0	$+\infty$	$-\infty$
Round away from 0	$+\infty$	$-\infty$
Round to prepare for shorter precision	$+N_{max}$	$-N_{max}$

Figure 18. Overflow Results When Exception Is Disabled

4. The result is placed into the target FPR
5. $FPSCR_{FR}$ is set to 1 if the returned result is $\pm\infty$, and is set to 0 if the returned result is $\pm N_{max}$
6. $FPSCR_{FI}$ is set to 1
7. $FPSCR_{FPRF}$ is set to indicate the class and sign of the result ($\pm\infty$ or \pm Normal number)

1.5.10.4 Underflow Exception

Definition

Except for *Reround*, the following describes the handling of the IEEE underflow exception condition. The *Reround* operation does not recognize an underflow exception condition.

The Underflow Exception is defined differently for the enabled and disabled states. However, a tininess condition is recognized in both states when a result computed as though both the precision and exponent range were unbounded would be nonzero and less than the target format's smallest normal number, N_{min} , in magnitude.

Unless otherwise defined in the instruction description, an underflow exception occurs as follows:

- Enabled:
When the tininess condition is recognized.
- Disabled:
When the tininess condition is recognized and when the delivered result value differs from what would have been computed were both the precision and the exponent range unbounded.

Action

The action to be taken depends on the setting of the Underflow Exception Enable bit of the FPSCR.

When Underflow Exception is enabled (FPSCR_{UE}=1) and underflow occurs, the following actions are taken:

1. Underflow Exception is set
FPSCR_{UX} ← 1
2. The infinitely precise result is multiplied by 10^α. That is, the exponent adjustment α is added to the exponent. This is called the *wrapped result*. The exponent adjustment for all operations, except for *Round To DFP32* and *Round To DFP64*, is 576 for DFP64 and 9216 for DFP128. For *Round To DFP32* and *Round To DFP64*, the exponent adjustment is 192 for the source format of DFP64 and 3072 for the source format of DFP128.
3. The wrapped result is rounded to the target-format precision. This is called the *wrapped rounded result*.
4. If the wrapped rounded result has only one form, it is the delivered result. If the wrapped rounded result has redundant forms and is exact, the result of the form that has the exponent closest to the wrapped ideal exponent is returned. If the wrapped rounded result has redundant forms and is inexact, the result of the form that has the smallest exponent is returned. The wrapped ideal exponent is the result of adding the exponent adjustment to the ideal exponent.
5. FPSCR_{FPRF} is set to indicate the class and sign of the result (± Normal number)

When Underflow Exception is disabled (FPSCR_{UE}=0) and underflow occurs, the following actions are taken:

1. Underflow Exception is set
FPSCR_{UX} ← 1
2. The infinitely precise result is rounded to the target-format precision.
3. The rounded result is returned. If this result has redundant forms, the result of the form that is closest to the ideal exponent is returned.
4. FPSCR_{FPRF} is set to indicate the class and sign of the result (± Normal number, ± Subnormal Number, or ± Zero)

1.5.10.5 Inexact Exception

Definition

Except for *Round to FP Integer Without Inexact*, the following describes the handling of the IEEE inexact exception condition. The *Round to FP Integer Without Inexact* does not recognize an inexact exception condition.

An Inexact Exception occurs when either of two conditions occur during rounding:

1. The delivered result differs from what would have been computed were both the precision and exponent range unbounded.
2. The rounded result overflows and Overflow Exception is disabled.

Action

The action to be taken does not depend on the setting of the Inexact Exception Enable bit of the FPSCR.

When Inexact Exception occurs, the following actions are taken:

1. Inexact Exception is set
FPSCR_{XX} ← 1
2. The rounded or overflowed result is placed into the target FPR
3. FPSCR_{FPRF} is set to indicate the class and sign of the result

Programming Note

In some implementations, enabling Inexact Exceptions may degrade performance more than does enabling other types of floating-point exception.

1.5.11 Summary of Normal Rounding And Range Actions

Figure 19 and Figure 20 summarize rounding and range actions, with the following exceptions:

- The *Reround* operation recognizes neither an underflow nor an overflow exception.

- The *Round to FP Integer Without Inexact* operation does not recognize the inexact operation exception.

Range of v	Case	Result (r) when Rounding Mode Is								
		RNE	RNTZ	RNAZ	RAFZ	RTMI	RFSP	RTPI	RTZ	
$v < -N_{max}, q < -N_{max}$	Overflow	$-\infty^1$	$-\infty^1$	$-\infty^1$	$-\infty^1$	$-\infty^1$	$-\infty^1$	-Nmax	-Nmax	-Nmax
$v < -N_{max}, q = -N_{max}$	Normal	-Nmax	-Nmax	-Nmax	—	—	—	-Nmax	-Nmax	-Nmax
$-N_{max} \leq v \leq -N_{min}$	Normal	b	b	b	b	b	b	b	b	b
$-N_{min} < v \leq -D_{min}$	Tiny	b*	b*	b*	b*	b*	b*	b*	b	b
$-D_{min} < v < -D_{min}/2$	Tiny	-Dmin	-Dmin	-Dmin	-Dmin	-Dmin	-Dmin	-Dmin	-0	-0
$v = -D_{min}/2$	Tiny	-0	-0	-Dmin	-Dmin	-Dmin	-Dmin	-Dmin	-0	-0
$-D_{min}/2 < v < 0$	Tiny	-0	-0	-0	-Dmin	-Dmin	-Dmin	-Dmin	-0	-0
$v = 0$	EZD	+0	+0	+0	+0	-0	+0	+0	+0	+0
$0 < v < +D_{min}/2$	Tiny	+0	+0	+0	+Dmin	+0	+Dmin	+Dmin	+Dmin	+0
$v = +D_{min}/2$	Tiny	+0	+0	+Dmin	+Dmin	+0	+Dmin	+Dmin	+Dmin	+0
$+D_{min}/2 < v < +D_{min}$	Tiny	+Dmin	+Dmin	+Dmin	+Dmin	+0	+Dmin	+Dmin	+Dmin	+0
$+D_{min} \leq v < +N_{min}$	Tiny	b*	b*	b*	b*	b	b*	b*	b*	b
$+N_{min} \leq v \leq +N_{max}$	Normal	b	b	b	b	b	b	b	b	b
$+N_{max} < v, q = +N_{max}$	Normal	+Nmax	+Nmax	+Nmax	—	+Nmax	+Nmax	—	—	+Nmax
$+N_{max} < v, q > +N_{max}$	Overflow	$+\infty^1$	$+\infty^1$	$+\infty^1$	$+\infty^1$	$+\infty^1$	$+\infty^1$	$+\infty^1$	$+\infty^1$	$+\infty^1$

Explanation:

- This situation cannot occur.
- 1 The normal result r is considered to have been incremented.
- * The rounded value, in the extreme case, may be Nmin. In this case, the exception conditions are underflow, inexact, and incremented.
- b The value derived when the precise result v is rounded to the destination's precision, including both bounded precision and bounded exponent range.
- q The value derived when the precise result v is rounded to the destination's precision, but assuming an unbounded exponent range.
- r This is the returned value when neither overflow nor underflow is enabled.
- v Precise result before rounding, assuming unbounded precision and an unbounded exponent range. For data-format conversion operations, v is the source value.
- Dmin Smallest (in magnitude) representable subnormal number in the target format.
- EZD The result r of the exact-zero-difference case applies only to ADD and SUBTRACT with both source operands having opposite signs. (For ADD and SUBTRACT, when both source operands have the same sign, the sign of the zero result is the same sign as the sign of the source operands.)
- Nmax Largest (in magnitude) representable finite number in the target format.
- Nmin Smallest (in magnitude) representable normalized number in the target format.
- RAFZ Round away from 0.
- RFSP Round to Prepare for Shorter Precision.
- RNAZ Round to Nearest, Ties away from 0.
- RNE Round to Nearest, Ties to even.
- RNTZ Round to Nearest, Ties toward 0.
- RTPI Round toward $+\infty$.
- RTMI Round toward $-\infty$.
- RTZ Round toward 0.

Figure 19. Rounding and Range Actions (Part 1)

Case	Is r inexact (r≠v)	Overflow exception enabled	Underflow exception enabled	Inexact exception enabled	Is r Incremented (r > v)	Is q inexact (q≠v)	Is q Incremented (q > v)	Returned Results and Status Setting*
Overflow	Yes ¹	No	—	No	No	—	—	T(r), OX← 1, FI← 1, FR← 0, XX ← 1
Overflow	Yes ¹	No	—	No	Yes	—	—	T(r), OX← 1, FI← 1, FR← 1, XX ← 1
Overflow	Yes ¹	No	—	Yes	No	—	—	T(r), OX← 1, FI← 1, FR← 0, XX ← 1, TX
Overflow	Yes ¹	No	—	Yes	Yes	—	—	T(r), OX← 1, FI← 1, FR← 1, XX ← 1, TX
Overflow	Yes ¹	Yes	—	—	—	No	No ¹	Tw(q≠β), OX← 1, FI← 0, FR← 0, TO
Overflow	Yes ¹	Yes	—	—	—	Yes	No	Tw(q≠β), OX← 1, FI← 1, FR← 0, XX← 1, TO
Overflow	Yes ¹	Yes	—	—	—	Yes	Yes	Tw(q≠β), OX← 1, FI← 1, FR← 1, XX← 1, TO
Normal	No	—	—	—	—	—	—	T(r), FI← 0, FR← 0
Normal	Yes	—	—	No	No	—	—	T(r), FI← 1, FR← 0, XX ← 1
Normal	Yes	—	—	No	Yes	—	—	T(r), FI← 1, FR← 1, XX ← 1
Normal	Yes	—	—	Yes	No	—	—	T(r), FI← 1, FR← 0, XX ← 1, TX
Normal	Yes	—	—	Yes	Yes	—	—	T(r), FI← 1, FR← 1, XX ← 1, TX
Tiny	No	—	No	—	—	—	—	T(r), FI← 0, FR← 0
Tiny	No	—	Yes	—	—	No ¹	No ¹	Tw(q•β), UX← 1, FI← 0, FR← 0, TU
Tiny	Yes	—	No	No	No	—	—	T(r), UX← 1, FI← 1, FR← 0, XX ← 1
Tiny	Yes	—	No	No	Yes	—	—	T(r), UX← 1, FI← 1, FR← 1, XX ← 1
Tiny	Yes	—	No	Yes	No	—	—	T(r), UX← 1, FI← 1, FR← 0, XX ← 1, TX
Tiny	Yes	—	No	Yes	Yes	—	—	T(r), UX← 1, FI← 1, FR← 1, XX ← 1, TX
Tiny	Yes	—	Yes	—	—	No	No ¹	Tw(q•β), UX← 1, FI← 0, FR← 0, TU
Tiny	Yes	—	Yes	—	—	Yes	No	Tw(q•β), UX← 1, FI← 1, FR← 0, XX ← 1, TU
Tiny	Yes	—	Yes	—	—	Yes	Yes	Tw(q•β), UX← 1, FI← 1, FR← 1, XX ← 1, TU

Explanation:

- The results do not depend on this condition.
- 1 This condition is true by virtue of the state of some condition to the left of this column.
- * Rounding sets only the FI and FR status flags. Setting of the OX, XX, or UX flag is part of the exception actions. They are listed here for reference.
- β Wrap adjust, which depends on the type of operation and operand format. For all operations except *Round to DFP32* and *Round to DFP64*, the wrap adjust depends on the target format: $\beta = 10^\alpha$, where α is 576 for DFP64, and 9216 for DFP128. For *Round to DFP32* and *Round to DFP64*, the wrap adjust depends on the source format: $\beta = 10^\kappa$ where κ is 192 for DFP64 and 3072 for DFP128.
- q The value derived when the precise result v is rounded to destination's precision, but assuming an unbounded exponent range.
- r The result as defined in Part 1 of this figure.
- v Precise result before rounding, assuming unbounded precision and unbounded exponent range.
- FI Floating-Point-Fraction-Inexact status flag, FPSCR_{FI}. This status flag is non-sticky.
- FR Floating-Point-Fraction-Rounded status flag, FPSCR_{FR}.
- OX Floating-Point Overflow Exception status flag, FPSCR_{OX}.
- TO The system floating-point enabled exception error handler is invoked for the overflow exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode.
- TU The system floating-point enabled exception error handler is invoked for the underflow exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode.
- TX The system floating-point enabled exception error handler is invoked for the inexact exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode.
- T(x) The value x is placed at the target operand location.
- Tw(x) The wrapped rounded result x is placed at the target operand location. For all operations except data format conversions, the wrapped rounded result is in the same format and length as normal results at the target location. For data format conversions, the wrapped rounded result is in the same format and length as the source, but rounded to the target-format precision.
- UX Floating-Point-Underflow-Exception status flag, FPSCR_{UX}
- XX Float-Point-Inexact-Exception Status flag, FPSCR_{XX}. The flag is a sticky version of FPSCR_{FI}. When FPSCR_{FI} is set to a new value, the new value of FPSCR_{XX} is set to the result of ORing the old value of FPSCR_{XX} with the new value of FPSCR_{FI}.

Figure 20. Rounding and Range Actions (Part 2)

1.6 DFP Instruction Descriptions

The following sections describe the DFP instructions. When a 128-bit operand is used, it is held in a FPR pair and the instruction mnemonic uses a letter “q” to mean the quad-precision operation. Note that in the following descriptions, FPXp denotes a FPR pair and must address an even-odd pair. If the FPXp field specifies an odd-numbered register, then the instruction form is invalid. The notation FPX[p] means either a FPR, FPX, or a FPR pair, FPXp.

For DFP instructions, if a DFP operand is returned, the CC field of the target operand is encoded using preferred DPD codes.

1.6.1 DFP Arithmetic Instructions

All DFP arithmetic instructions are X-form instructions. They all set the FI and FR status flags, and also set the FPSCR_{FPRF} field. Furthermore, they all have an ideal exponent assigned and have the record bit provided.

The arithmetic instructions consist of Add, Divide, Multiply, and Subtract.

DFP Add [Quad]

X-form

dadd FRT,FRA,FRB (Rc=0)
dadd. FRT,FRA,FRB (Rc=1)

59	FRT	FRA	FRB	2	Rc
0	6	11	16	21	31

daddq FRTp,FRAp,FRBp (Rc=0)
daddq. FRTp,FRAp,FRBp (Rc=1)

63	FRTp	FRAp	FRBp	2	Rc
0	6	11	16	21	31

The DFP operand in register FRA[p] is added to the DFP operand in register FRB[p].

The result is rounded to the target-format precision under control of the DRN (bits 29:31) of the FPSCR. An appropriate form of the rounded result is selected based on the ideal exponent and is placed in the target register FRT[p]. The ideal exponent is the smaller exponent of the two source operands.

The following table summarizes the actions for Add. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation exception, in which case the field remains unchanged.

Special Registers Altered:

FPRF FR FI
FX OX UX XX
VXSNAN VXISI
CR1 (if Rc=1)

DFP Subtract [Quad]

X-form

dsub FRT,FRA,FRB (Rc=0)
dsub. FRT,FRA,FRB (Rc=1)

59	FRT	FRA	FRB	514	Rc
0	6	11	16	21	31

dsubq FRTp,FRAp,FRBp (Rc=0)
dsubq. FRTp,FRAp,FRBp (Rc=1)

63	FRTp	FRAp	FRBp	514	Rc
0	6	11	16	21	31

The DFP operand in register FRB[p] is subtracted from the DFP operand in register FRA[p]. The result is rounded to the target-format precision under control of the DRN (bits 29:31) of the FPSCR. An appropriate form of the rounded result is selected based on the ideal exponent and is placed in the target register FRT[p]. The ideal exponent is the smaller exponent of the two source operands.

The execution of Subtract is identical to that of Add, except that the operand in FRB participates in the operation with its sign bit inverted. See the actions table for Add.

Special Registers Altered:

FPRF FR FI
FX OX UX XX
VXSNAN VXISI
CR1 (if Rc=1)

Operand a in FRA[p] is	Actions for Add (a + b) when operand b in FRB[p] is				
	$-\infty$	F	$+\infty$	QNaN	SNaN
$-\infty$	T(-dINF)	T(-dINF)	V _{XISI} : T(dNaN)	P(b)	V _{XSNAN} : U(b)
F	T(-dINF)	S(a + b)	T(+dINF)	P(b)	V _{XSNAN} : U(b)
$+\infty$	V _{XISI} : T(dNaN)	T(+dINF)	T(+dINF)	P(b)	V _{XSNAN} : U(b)
QNaN	P(a)	P(a)	P(a)	P(a)	V _{XSNAN} : U(b)
SNaN	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)

Explanation:

a + b The value a added to b, rounded to the target-format precision and returned in the appropriate form. (See Section 1.5.11 on page 21)

+dINF Default plus infinity.

-dINF Default minus infinity.

dNaN Default quiet NaN.

F All finite numbers, including zeros.

P(x) The QNaN of operand x is propagated and placed in FRT[p].

S(x) The value x is placed in FRT[p] with the sign set by the rules of algebra. When the source operands have the same sign, the sign of the result is the same as the sign of the operands, including the case when the result is zero. When the operands have opposite signs, the sign of a zero result is positive in all rounding modes, except round toward $-\infty$, in which case, the sign is minus.

T(x) The value x is placed in FRT[p].

U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FRT[p].

V_{XISI} The Invalid-Operation Exception (VXISI) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 “Invalid Operation Exception” on page 18 for the exception actions.)

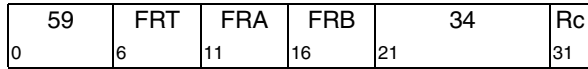
V_{XSNAN} The Invalid-Operation Exception (VXSNAN) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 “Invalid Operation Exception” on page 18 for the exception actions.)

Figure 21. Actions: Add

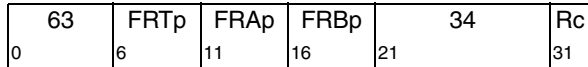
DFP Multiply [Quad]

X-form

dmul FRT,FRA,FRB (Rc=0)
 dmul. FRT,FRA,FRB (Rc=1)



dmulq FRTp,FRAp,FRBp (Rc=0)
 dmulq. FRTp,FRAp,FRBp (Rc=1)



The DFP operand in register FRA[p] is multiplied by the DFP operand in register FRB[p].

The result is rounded to the target-format precision under control of the DRN (bits 29:31) of the FPSCR. An appropriate form of the rounded result is selected based on the ideal exponent and is placed in the target register FRT[p]. The ideal exponent is the sum of the two exponents of the source operands.

The following table summarizes the actions for Multiply. The table does not include the setting of the FPSCR_{F_{PRF}} field. The FPSCR_{F_{PRF}} field is always set to the class and sign of the result, except for an enabled invalid-operation exception, in which case the field remains unchanged.

Special Registers Altered:

- FPRF FR FI
- FX OX UX XX
- VXSNAN VXIMZ
- CR1 (if Rc=1)

Operand a in FRA[p] is	Actions for Multiply (a*b) when operand b in FRB[p] is				
	0	Fn	∞	QNaN	SNaN
0	S(a * b)	S(a * b)	V _{XIMZ} : T(dNaN)	P(b)	V _{XSNAN} : U(b)
Fn	S(a * b)	S(a * b)	S(dINF)	P(b)	V _{XSNAN} : U(b)
∞	V _{XIMZ} : T(dNaN)	S(dINF)	S(dINF)	P(b)	V _{XSNAN} : U(b)
QNaN	P(a)	P(a)	P(a)	P(a)	V _{XSNAN} : U(b)
SNaN	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)

Explanation:

- a * b The value a multiplied by b, rounded to the target-format precision and returned in the appropriate form. (See Section 1.5.11 on page 21)
- dINF Default infinity.
- dNaN Default quiet NaN.
- Fn Finite nonzero number (includes both normal and subnormal numbers).
- P(x) The QNaN of operand x is propagated and placed in FRT[p].
- S(x) The value x is placed in FRT[p] with the sign set to the exclusive-OR of the source-operand signs.
- T(x) The value x is placed in FRT[p].
- U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FRT[p].
- V_{XIMZ}: The Invalid-Operation Exception (VXIMZ) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 "Invalid Operation Exception" on page 18 for the exception actions.)
- V_{XSNAN}: The Invalid-Operation Exception (VXSNAN) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 "Invalid Operation Exception" on page 18 for the exception actions.)

Figure 22. Actions: Multiply

DFP Divide [Quad]

X-form

ddiv FRT,FRA,FRB (Rc=0)
 ddiv. FRT,FRA,FRB (Rc=1)

59	FRT	FRA	FRB	546	Rc
0	6	11	16	21	31

ddivq FRTp,FRAp,FRBp (Rc=0)
 ddivq. FRTp,FRAp,FRBp (Rc=1)

63	FRTp	FRAp	FRBp	546	Rc
0	6	11	16	21	31

The DFP operand in register FRA[p] is divided by the DFP operand in register FRB[p].

The result is rounded to the target-format precision under control of the DRN (bits 29:31) of the FPSCR.

An appropriate form of the rounded result is selected based on the ideal exponent and is placed in the target register FRT[p]. The ideal exponent is the difference of subtracting the exponent of the divisor from the exponent of the dividend.

The following table summarizes the actions for Divide. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation and enabled zero-divide exceptions, in which cases the field remains unchanged.

Special Registers Altered:

FPRF FR FI
 FX OX UX ZX XX
 VXSNaN VXIDI VXZDZ
 CR1 (if Rc=1)

Operand a in FRA[p] is	Actions for Divide (a ÷ b) when operand b in FRB[p] is				
	0	Fn	∞	QNaN	SNaN
0	V _{XZDZ} : T(dNaN)	S(a ÷ b)	S(zt)	P(b)	V _{XSNAN} : U(b)
Fn	Zx: S(dINF)	S(a ÷ b)	S(zt)	P(b)	V _{XSNAN} : U(b)
∞	S(dINF)	S(dINF)	V _{XIDI} : T(dNaN)	P(b)	V _{XSNAN} : U(b)
QNaN	P(a)	P(a)	P(a)	P(a)	V _{XSNAN} : U(b)
SNaN	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)	V _{XSNAN} : U(a)

Explanation:

- a ÷ b The value a divided by b, rounded to the target-format precision and returned in the appropriate form. (See Section 1.5.11 on page 21.)
- dINF Default infinity.
- dNaN Default quiet NaN.
- Fn Finite nonzero number (includes both normal and subnormal numbers).
- P(x) The QNaN of operand x is propagated and placed in FRT[p].
- S(x) The value x is placed in FRT[p] with the sign set to the exclusive-OR of the source-operand signs.
- T(x) The value x is placed in FRT[p].
- U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FRT[p].
- V_{XIDI}: The Invalid-Operation Exception (VXIDI) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 “Invalid Operation Exception” on page 18 for the exception actions.)
- V_{XSNAN}: The Invalid-Operation Exception (VXSNaN) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 “Invalid Operation Exception” on page 18 for the exception actions.)
- V_{XZDZ}: The Invalid-Operation Exception (VXZDZ) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 “Invalid Operation Exception” on page 18 for the exception actions.)
- zt True zero (zero coefficient and most negative exponent).
- Zx The Zero-Divide Exception occurs. The result is produced only when the exception is disabled (See Section 1.5.10.2 “Zero Divide Exception” on page 18 for the exception actions.)

Figure 23. Actions: Divide

1.6.2 DFP Compare Instructions

The DFP compare instructions consist of the *Compare Ordered* and *Compare Unordered* instructions. The compare instructions do not provide the record bit.

The comparison sets the designated CR field to indicate the result. The FPSCR_{FPCC} is set in the same way.

The codes in the CR field BF and FPSCR_{FPCC} are defined for the DFP compare operations as follows.

Bit	Name	Description
-----	------	-------------

0	FL	(FRA[p]) < (FRB[p])
1	FG	(FRA[p]) > (FRB[p])
2	FE	(FRA[p]) = (FRB[p])
3	FU	(FRA[p]) ? (FRB[p])

DFP Compare Unordered [Quad] X-form

dcmpu BF,FRA,FRB

0	59	BF	//	FRA	FRB	642	/
	6	9	11	16	21		31

dcmpuq BF,FRAp,FRBp

0	63	BF	//	FRAp	FRBp	642	/
	6	9	11	16	21		31

The DFP operand in register FRA[p] is compared to the DFP operand in register FRB[p]. The result of the compare is placed into CR field BF and the FPSCR_{FPCC}.

Special Registers Altered:

- CR field BF
- FPCC
- FX
- VXSNAN

Operand a in FRA[p] is	Actions for Compare Unordered (a:b) when operand b in FRB[p] is				
	$-\infty$	F	$+\infty$	QNaN	SNaN
$-\infty$	Fe	FI	FI	Fu	Fu, VXSNAN
F	Fg	C(a:b)	FI	Fu	Fu, VXSNAN
$+\infty$	Fg	Fg	Fe	Fu	Fu, VXSNAN
QNaN	Fu	Fu	Fu	Fu	Fu, VXSNAN
SNaN	Fu, VXSNAN	Fu, VXSNAN	Fu, VXSNAN	Fu, VXSNAN	Fu, VXSNAN

Explanation:

- C(a:b) Algebraic comparison. See the table below.
- F All finite numbers, including zeros.
- Fe Set the 0b0010 code in CR field BF and FPSCR_{FPCC}.
- Fg Set the 0b0100 code in CR field BF and FPSCR_{FPCC}.
- FI Set the 0b1000 code in CR field BF and FPSCR_{FPCC}.
- Fu Set the 0b0001 code in CR field BF and FPSCR_{FPCC}.
- V_{XSNAN} The invalid-operation exception (VXSNAN) occurs. See Section 1.5.10.1 for actions.

Relation of Value a to Value b	Action for C(a:b)
a = b	Fe
a < b	FI
a > b	Fg

Figure 24. Actions: Compare Unordered

DFP Compare Ordered [Quad] X-form

dcmpo BF,FRA,FRB

59	BF	//	FRA	FRB	130	/
0	6	9	11	16	21	31

dcmpoq BF,FRAp,FRBp

63	BF	//	FRAp	FRBp	130	/
0	6	9	11	16	21	31

The DFP operand in register FRA[p] is compared to the DFP operand in register FRB[p]. The result of the compare is placed into CR field BF and the FPSCR_{FPCC}.

Special Registers Altered:

- CR field BF
- FPCC
- FX
- VXSNAN VXVC

Operand a in FRA[p] is	Actions for Compare ordered (a:b) when operand b in FRB[p] is				
	$-\infty$	F	$+\infty$	QNaN	SNaN
$-\infty$	Fe	FI	FI	Fu, V _{XVC}	Fu, V _{XSV}
F	Fg	C(a:b)	FI	Fu, V _{XVC}	Fu, V _{XSV}
$+\infty$	Fg	Fg	Fe	Fu, V _{XVC}	Fu, V _{XSV}
QNaN	Fu, V _{XVC}	Fu, V _{XVC}	Fu, V _{XVC}	Fu, V _{XVC}	Fu, V _{XSV}
SNaN	Fu, V _{XSV}	Fu, V _{XSV}	Fu, V _{XSV}	Fu, V _{XSV}	Fu, V _{XSV}

Explanation:

- C(a:b) Algebraic comparison. See the table below
- F All finite numbers, including zeros
- Fe Set the 0b0010 code in CR field BF and FPSCR_{FPCC}
- Fg Set the 0b0100 code in CR field BF and FPSCR_{FPCC}
- FI Set the 0b1000 code in CR field BF and FPSCR_{FPCC}
- Fu Set the 0b0001 code in CR field BF and FPSCR_{FPCC}
- V_{XSV} The invalid-operation exception (VXSNAN) occurs. Additionally, if the exception is disabled (FPSCR_{VE}=0), then FPSCR_{VXVC} is also set to one. See Section 1.5.10.1 for actions.
- V_{XVC} The invalid-operation exception (VXVC) occurs. See Section 1.5.10.1 for actions.

Relation of Value a to Value b	Action for C(a:b)
a = b	Fe
a < b	FI
a > b	Fg

Figure 25. Actions: Compare Ordered

1.6.3 DFP Test Instructions

The DFP test instructions consist of the *Test Data Class*, *Test Data Group*, *Test Exponent*, and *Test Significance* instructions, and they do not provide the record bit.

The test instructions set the designated CR field to indicate the result. The $FPSCR_{FPCC}$ is set in the same way.

DFP Test Data Class [Quad] Z-form

dtstdc BF,FRA,DCM

0	59	BF	//	FRA	DCM	194	/
		6	9	11	16	22	31

dtstdcq BF,FRAp,DCM

0	63	BF	//	FRAp	DCM	194	/
		6	9	11	16	22	31

The DFP operand in register FRA[p] is classified into one of 6 data classes to select one bit from the DCM (Data Class Mask) field. The test result is placed in CR field BF and $FPSCR_{FPCC}$ to indicate whether the selected bit in the DCM field is one or zero, and whether the sign of the source operand is plus or minus. Bits in the DCM field define the data classes as shown below.

DCM Bit	Data Class
0	Zero
1	Subnormal
2	Normal
3	Infinity
4	Quiet NaN
5	Signaling NaN

The codes in CR field BF and FPCC are defined as follows.

Field	Meaning
0000	Operand positive with no match
0010	Operand positive with match
1000	Operand negative with no match
1010	Operand negative with match

Special Registers Altered:

CR field BF
FPCC

DFP Test Data Group [Quad] Z-form

dtstdg BF,FRA,DGM

0	59	BF	//	FRA	DGM	226	/
		6	9	11	16	22	31

dtstdgq BF,FRAp,DGM

0	63	BF	//	FRAp	DGM	226	/
		6	9	11	16	22	31

The DFP operand in register FRA[p] is classified into one of 6 data groups to select one bit from the DGM (Data Group Mask) field. The test result is placed in CR field BF and $FPSCR_{FPCC}$ to indicate whether the selected bit in the DGM field is one or zero, and whether the sign of the source operand is plus or minus. Bits in the DGM field define the data groups as shown below. The term extreme exponent means either the maximum exponent, X_{max} , or the minimum exponent, X_{min} .

DGM Bit	Data Group
0	Zero with non-extreme exponent
1	Zero with extreme exponent
2	Subnormal or (Normal with extreme exponent)
3	Normal with non-extreme exponent and leftmost zero digit in coefficient
4	Normal with non-extreme exponent and leftmost nonzero digit in coefficient
5	Special symbol (Infinity, QNaN, or SNaN)

The codes in CR field BF and FPCC are defined as follows.

Field	Meaning
0000	Operand positive with no match
0010	Operand positive with match
1000	Operand negative with no match
1010	Operand negative with match

Special Registers Altered

CR field BF
FPCC

DFP Test Exponent [Quad] X-form

dtstex BF,FRA,FRB

59	BF	//	FRA	FRB	162	/
0	6	9	11	16	21	31

dtstexq BF,FRAp,FRBp

63	BF	//	FRAp	FRBp	162	/
0	6	9	11	16	21	31

The exponent value (Ea) of the DFP operand in FRA[p] is compared to the exponent value (Eb) of the DFP operand in FRB [p]. The result of the compare is placed into CR field BF and the FPSCR_{FPCC}.

The codes in the CR field BF and FPSCR_{FPCC} are defined for the *DFP Test Exponent* operations as follows.

Bit	Description
0	Ea < Eb
1	Ea > Eb
2	Ea = Eb
3	Ea ? Eb

Special Registers Altered:

CR field BF
FPCC

Operand a in FRA[p] is	Actions for Test Exponent (Ea:Eb) when operand b in FRB[p] is			
	F	∞	QNaN	SNaN
F	C(Ea:Eb)	Fu	Fu	Fu
∞	Fu	Fe	Fu	Fu
QNaN	Fu	Fu	Fe	Fe
SNaN	Fu	Fu	Fe	Fe

Explanation:

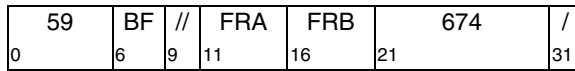
C(Ea:Eb)	Algebraic comparison. See the table below.
F	All finite numbers, including zeros.
Fe	Set the 0b0010 code in CR field BF and FPSCR _{FPCC} .
Fg	Set the 0b0100 code in CR field BF and FPSCR _{FPCC} .
Fl	Set the 0b1000 code in CR field BF and FPSCR _{FPCC} .
Fu	Set the 0b0001 code in CR field BF and FPSCR _{FPCC} .

Relation of Value Ea to Value Eb	Action for C(Ea:Eb)
Ea = Eb	Fe
Ea < Eb	Fl
Ea > Eb	Fg

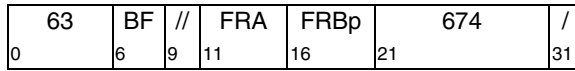
Figure 26. Actions: Test Exponent

DFP Test Significance [Quad] X-form

dtstsf BF,FRA,FRB

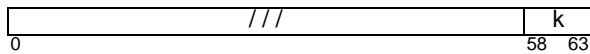


dtstsfq BF,FRA,FRBp



The number of significant digits of the DFP operand in FRB[p], NSDb, is compared to the referenced significance, k. For this instruction, the number of significant digits of the value 0 is considered to be zero. The result of the compare is placed into CR field BF and the FPSCR_{FPCC}.

The source operand in FRA contains a 6-bit unsigned binary integer that specifies the referenced significance. The format of the source operand in FRA is defined below.



The codes in the CR field BF and FPSCR_{FPCC} are defined for the *DFP Test Significance* operation as follows.

Bit	Description
0	k ≠ 0 and k < NSDb
1	k ≠ 0 and k > NSDb, or k = 0
2	k ≠ 0 and k = NSDb
3	k ? NSDb

Special Registers Altered:

- CR field BF
- FPCC

Actions for Test Significance when the operand in FRB[p] is			
F	∞	QNaN	SNaN
C(k: NSDb)	Fu	Fu	Fu
Explanation:			
C(k: NSDb)	Algebraic comparison. See the table below.		
F	All finite numbers, including zeros.		
Fe	Set the 0b0010 code in CR field BF and FPSCR _{FPCC} .		
Fg	Set the 0b0100 code in CR field BF and FPSCR _{FPCC} .		
FI	Set the 0b1000 code in CR field BF and FPSCR _{FPCC} .		
Fu	Set the 0b0001 code in CR field BF and FPSCR _{FPCC} .		

Relation of Value NSDb to Value k	Action for C(k:NSDb)
k ≠ 0 and k = NSDb	Fe
k ≠ 0 and k < NSDb	FI
k ≠ 0 and k > NSDb, or k = 0	Fg

Figure 27. Actions: Test Significance

1.6.4 DFP Quantum Adjustment Instructions

The *Quantum Adjustment* operations consist of the *Quantize*, *Quantize Immediate*, *Reround*, and *Round To FP Integer* operations.

The *Quantum Adjustment* instructions are Z-form instructions and have an immediate RMC (Rounding-Mode-Control) field, which specifies the rounding mode used. For *Quantize*, *Quantize Immediate*, and *Reround*, the RMC field contains the primary encoding. For *Round to FP Integer*, the field contains either pri-

mary or secondary encoding, depending on the setting of a RMC-encoding-selection bit. See Section 1.5.2 “Rounding Mode Specification” on page 13 for the definition of RMC encoding.

All *Quantum Adjustment* instructions set the FI and FR status flags, and also set the FPSCR_{FPRF} field. The record bit is provided to each of these instructions. They return the target operand in a form with the ideal exponent.

DFP Quantize Immediate [Quad] Z-form

dquai TE,FRT,FRB,RMC (Rc=0)
 dquai. TE,FRT,FRB,RMC (Rc=1)

59	FRT	TE	FRB	RMC	67	Rc
0	6	11	16	21	23	31

dquaiq TE,FRTp,FRBp,RMC (Rc=0)
 dquaiq. TE,FRTp,FRBp,RMC (Rc=1)

63	FRTp	TE	FRBp	RMC	67	Rc
0	6	11	16	21	23	31

The DFP operand in the source register FRB[p] is converted and rounded to the form with the exponent specified by TE based on the rounding mode specified in the RMC field. TE is a 5-bit signed binary integer. The result of that form is placed in the target register FRT[p]. The sign of the result is the same as the sign of the source operand in FRB[p]. The ideal exponent is the exponent specified by TE.

When the value of the source operand in FRB[p] is greater than $(10^{p-1}) \times 10^{TE}$, where p is the format precision, an invalid operation exception is recognized.

When the delivered result differs in value from the source operand in FRB[p], an inexact exception is recognized. No underflow exception is recognized by this operation, regardless of the value of the source operand in FRB[p].

Figure 28 and Figure 29 summarize the actions. The tables do not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation exception, in which case the field remains unchanged.

Special Registers Altered:

- FPRF FR FI
- FX XX
- VXSNAN VXCVI
- CR1 (if Rc=1)

DFP Quantize [Quad] Z-form

dqua FRT,FRA,FRB,RMC (Rc=0)
 dqua. FRT,FRA,FRB,RMC (Rc=1)

59	FRT	FRA	FRB	RMC	3	Rc
0	6	11	16	21	23	31

dquaq FRTp,FRAp,FRBp,RMC (Rc=0)
 dquaq. FRTp,FRAp,FRBp,RMC (Rc=1)

63	FRTp	FRAp	FRBp	RMC	3	Rc
0	6	11	16	21	23	31

The DFP operand in the source register FRB[p] is converted and rounded to the form with the same exponent as that of the DFP operand in FRA[p] based on the rounding mode specified in the RMC field. The result of that form is placed in the target register FRT[p]. The sign of the result is the same as the sign of the source operand in FRB[p]. The ideal exponent is the exponent specified in FRA[p].

When the value of the source operand in FRB[p] is greater than $(10^{p-1}) \times 10^{Ea}$, where p is the format precision and Ea is the exponent of the operand in FRA[p], an invalid operation exception is recognized.

When the delivered result differs in value from the source operand in FRB[p], an inexact exception is recognized. No underflow exception is recognized by this operation, regardless of the value of the source operand in FRB[p].

The following tables summarize the actions. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation exception, in which case the field remains unchanged.

Special Register Altered:

- FPRF FR FI
- FX XX
- VXSNAN VXCVI
- CR1 (if Rc=1)

Operand a in FRA[p] is	Actions for Quantize when operand b in FRB[p] is				
	0	Fn	∞	QNaN	SNaN
0	*	*	V_{XCVI} : T(dNaN)	P(b)	V_{XSNAN} : U(b)
Fn	*	*	V_{XCVI} : T(dNaN)	P(b)	V_{XSNAN} : U(b)
•	V_{XCVI} : T(dNaN)	V_{XCVI} : T(dNaN)	T(dINF)	P(b)	V_{XSNAN} : U(b)
QNaN	P(a)	P(a)	P(a)	P(a)	V_{XSNAN} : U(b)
SNaN	V_{XSNAN} : U(a)	V_{XSNAN} : U(a)	V_{XSNAN} : U(a)	V_{XSNAN} : U(a)	V_{XSNAN} : U(a)

Explanation:

- * See next table.
- dINF Default infinity
- dNaN Default quiet NaN
- Fn Finite nonzero numbers (includes both subnormal and normal numbers)
- P(x) The QNaN of operand x is propagated and placed in FRT[p]
- T(x) The value x is placed in FRT[p]
- U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FRT[p].
- V_{XCVI} The Invalid-Operation Exception (VXCVI) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 for actions)
- V_{XSNAN} The Invalid-Operation Exception (VXSNAN) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 for actions)

Figure 28. Actions (part 1) Quantize

	Actions for Quantize when operand b in FRB[p] is		
		0	Fn
Te < Se	$V_b > (10^p - 1) \times 10^{Te}$	E(0)	V_{XCVI} : T(dNaN)
	$V_b \leq (10^p - 1) \times 10^{Te}$	E(0)	L(b)
Te = Se		E(0)	W(b)
Te > Se		E(0)	QR(b)

Explanation:

- dNaN Default quiet NaN
- E(0) The value of zero with the exponent value Te is placed in FRT[p].
- L(x) The operand x is converted to the form with the exponent value Te.
- p The precision of the format.
- QR(x) The operand x is rounded to the result of the form with the exponent value Te based on the specified rounding mode. The result of that form is placed in FRT[p].
- Se The exponent of the source operand in FRB[p].
- Te The target exponent; FRA[p] for *dqua[q]*, or TE, a 5-bit signed binary integer for *dqua[q]*.
- T(x) The value x is placed in FRT[p].
- V_b The value of the source operand in FRB[p].
- W(x) The value and the form of operand x is placed in FRT[p].
- V_{XCVI} : The Invalid-Operation Exception (VXCVI) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 for actions.)

Figure 29. Actions (part2) Quantize

DFP Reround [Quad] Z-form

drrnd FRT,FRA,FRB,RMC (Rc=0)
 drrnd. FRT,FRA,FRB,RMC (Rc=1)

59	FRT	FRA	FRB	RMC	35	Rc
0	6	11	16	21	23	31

drrndq FRTp,FRA,FRBp,RMC (Rc=0)
 drrndq. FRTp,FRA,FRBp,RMC (Rc=1)

63	FRTp	FRA	FRBp	RMC	35	Rc
0	6	11	16	21	23	31

When the DFP operand in the source register FRB[p] is a finite number, and if the referenced significance is zero, or if the referenced significance is nonzero and the number of significant digits of the source operand is less than or equal to the referenced significance, then the value and the form of the source operand is placed in the target register FRT[p]. If the referenced significance is nonzero and the number of significant digits of the source operand is greater than the referenced significance, then the source operand is converted and rounded to the number of significant digits specified in the referenced significance based on the rounding mode specified in the RMC field. The result of the form with the specified number of significant digits is placed in the target register FRT[p]. The sign of the result is the same as the sign of the source operand in FRB[p].

For this instruction, the number of significant digits of the value 0 is considered to be zero. The ideal exponent is the greater value of the exponent of the source operand in FRB[p] and the referenced exponent. The referenced exponent is the resultant exponent if the source operand in FRB[p] would have been converted and rounded to the number of significant digits specified in the referenced significance based on the rounding mode specified in the RMC field.

The referenced-significance field of the source operand in FRA contains a 6-bit unsigned binary integer that specifies the number of significant digits of the target operand. The format of the source operand in FRA is defined below.

///					k
0					58 63

If the exponent of the rounded result of the form that has the specified number of significant digits would be greater than X_{max} , an invalid operation exception (VXCVI) occurs. When the invalid-operation exception occurs, and if the exception is disabled, a default QNaN is returned. When an invalid-operation exception occurs, no inexact exception is recognized.

In the absence of an invalid-operation exception, if the result differs in value from the source operand in FRB[p], an inexact exception is recognized.

This operation causes neither an overflow nor an underflow exception.

The following table summarizes the actions for *Reround*. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation exception, in which case the field remains unchanged.

Special Registers Altered:

- FPRF FR FI
- FX XX
- VXSNAN VXCVI
- CR1

(if Rc=1)

	Actions for Reround when operand b in FRB[p] is				
	0*	Fn	∞	QNaN	SNaN
k \neq 0, k < m	-	RR(b) or V _{XCVI} : T(dNaN)	T(dINF)	P(b)	V _{XSNAN} : U(b)
k \neq 0, k = m	-	W(b)	T(dINF)	P(b)	V _{XSNAN} : U(b)
k \neq 0 and k > m, or k = 0	W(b)	W(b)	T(dINF)	P(b)	V _{XSNAN} : U(b)

Explanation:

- * The number of significant digits of the value 0 is considered to be zero for this instruction.
- Not applicable.
- dINF Default infinity.
- Fn Finite nonzero numbers (includes both subnormal and normal numbers).
- k Referenced significance, which specifies the number of significant digits in the target operand.
- m Number of significant digits in the source operand in FRB[p].
- P(x) The QNaN of operand x is propagated and placed in FRT[p].
- RR(x) The value x is rounded to the form that has the specified number of significant digits.
If $RR(x) \leq (10^k - 1) \times 10^{x_{max}}$, then RR(x) is returned; otherwise an invalid-operation exception is recognized.
- T(x) The value x is placed in FRT[p].
- U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FRT[p].
- V_{XCVI} The Invalid-Operation Exception (VXCVI) occurs. The result is produced only when the exception is disabled. (See Section 1.5.10.1 for actions.)
- V_{XSNAN}: The Invalid-Operation Exception (VXSNAN) occurs. The result is produced only when the exception is disabled. See Section 1.5.10.1 for actions.
- W(x) The value and the form of x is placed in FRT[p].

Figure 30. Actions: Reround

DFP Round To FP Integer With Inexact Z-form

drintx R,FRT,FRB,RMC (Rc=0)
 drintx. R,FRT,FRB,RMC (Rc=1)

59	FRT	//	R	FRB	RMC	99	Rc
0	6	11	15	16	21	23	31

drintxq FRTp,FRBp,RMC (Rc=0)
 drintxq. FRTp,FRBp,RMC (Rc=1)

63	FRTp	//	R	FRBp	RMC	99	Rc
0	6	11	15	16	21	23	31

The DFP operand in the source register FRB[p] is converted and rounded to a floating-point integer. The rounded value in the form of a floating-point integer is placed into the target register FRT[p] in the same source format. The sign of the result is the same as the sign of the source operand. The ideal exponent is the larger value of zero and the exponent of the source operand in FRB[p].

The rounding mode used is specified in the RMC field. When the RMC-encoding-selection (R) bit is zero, the RMC field contains the primary encoding; when the bit is one, the field contains the secondary encoding.

In addition to coercion of the converted value to fit the target format, the special rounding used by *Round To FP Integer* also coerces the target exponent to the ideal exponent.

When the source operand is a finite number, and if the source exponent is less than zero, the source operand is converted and rounded to the result with the target exponent of zero. When the source exponent is greater than or equal to zero, the result is set to the numerical value and the form of the source operand.

When the result differs in value from the source operand in FRB[p], an inexact exception is recognized. No underflow exception is recognized by this operation, regardless of the value of the source operand in FRB[p].

The following table summarizes the actions for *Round To FP Integer With Inexact*. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation, in which case the field remains unchanged.

Special Registers Altered:

FPRF FR FI
 FX XX
 VXSNaN
 CR1 (if Rc=1)

Operand b in FRB is	Is n not precise ($n \neq b$)	Inv.-Op. Exception Enabled	Inexact Exception Enabled	Is n Incremented ($ n > b $)	Actions*
$-\infty$	No ¹	-	-	-	T(-dINF), FI \leftarrow 0, FR \leftarrow 0
F	No	-	-	-	W(n), FI \leftarrow 0, FR \leftarrow 0
F	Yes	-	No	No	W(n), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1
F	Yes	-	No	Yes	W(n), FI \leftarrow 1, FR \leftarrow 1, XX \leftarrow 1
F	Yes	-	Yes	No	W(n), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1, TX
F	Yes	-	Yes	Yes	W(n), FI \leftarrow 1, FR \leftarrow 1, XX \leftarrow 1, TX
$+\infty$	No ¹	-	-	-	T(+dINF), FI \leftarrow 0, FR \leftarrow 0
QNaN	No ¹	-	-	-	P(b), FI \leftarrow 0, FR \leftarrow 0
SNaN	No ¹	No	-	-	U(b), FI \leftarrow 0, FR \leftarrow 0, VXSNaN \leftarrow 1
SNaN	No ¹	Yes	-	-	VXSNaN \leftarrow 1, TV
<p>Explanation:</p> <ul style="list-style-type: none"> * Setting of XX and VXSNaN is part of the corresponding exception actions. Also, when an invalid-operation exception occurs, setting of FI and FR is part of the exception actions. (See the sections, “Inexact Exception” and “Invalid Operation Exception” for more details.) - The actions do not depend on this condition. ¹ This condition is true by virtue of the state of some condition to the left of this column. dINF Default infinity. F All finite numbers, including zeros. FI Floating-Point-Fraction-Inexact status flag, FPSCR_{FI}. FR Floating-Point-Fraction-Rounded status flag, FPSCR_{FR}. n The value derived when the source operand, b, is rounded to an integer using the special rounding for <i>Round To FP Integer</i>. P(x) The QNaN of operand x is propagated and placed in FRT[p]. T(x) The value x is placed in FRT[p]. TV The system floating-point enabled exception error handler is invoked for the invalid-operation exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode. TX The system floating-point enabled exception error handler is invoked for the inexact exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode. U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FPT[p]. W(x) The value x in the form of zero exponent or the source exponent is placed in FRT[p]. XX Floating-Point-Inexact-Exception status flag, FPSCR_{XX}. 					

Figure 31. Actions: Round to FP Integer With Inexact

DFP Round To FP Integer Without Inexact [Quad] Z-form

Special Registers Altered:

drintrn R,FRT,FRB,RMC (Rc=0)
 drintrn. R,FRT,FRB,RMC (Rc=1)

FPRF FR FI (FR and FI are set to zeros)
 FX
 VXSNaN
 CR1 (if Rc=1)

59	FRT	//	R	FRB	RMC	227	Rc
0	6	11	15	16	21	23	31

drintrq FRTp,FRBp,RMC (Rc=0)
 drintrq. FRTp,FRBp,RMC (Rc=1)

63	FRTp	//	R	FRBp	RMC	227	Rc
0	6	11	15	16	21	23	31

This operation is the same as the *Round To FP Integer With Inexact* operation, except that this operation does not recognize an inexact exception.

The following table summarizes the actions for *Round To FP Integer Without Inexact*. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result, except for an enabled invalid-operation, in which case the field remains unchanged.

Operand b in FRB is	Inv.-Op. Exception Enabled	Actions*
$-\infty$	-	T(-dINF), FI \leftarrow 0, FR \leftarrow 0
F	-	W(n), FI \leftarrow 0, FR \leftarrow 0
$+\infty$	-	T(+dINF), FI \leftarrow 0, FR \leftarrow 0
QNaN	-	P(b), FI \leftarrow 0, FR \leftarrow 0
SNaN	No	U(b), FI \leftarrow 0, FR \leftarrow 0, VXSNaN \leftarrow 1
SNaN	Yes	VXSNaN \leftarrow 1, TV

Explanation:

- * Setting of VXSNaN is part of the corresponding exception actions. Also, when an invalid-operation exception occurs, setting of FI and FR bits is part of the exception actions. (See the sections, "Invalid Operation Exception" for more details.)
- The actions do not depend on this condition.
- dINF Default infinity.
- F All finite numbers, including zeros.
- FI Floating-Point-Fraction-Inexact status flag, FPSCR_{FI}.
- FR Floating-Point-Fraction-Rounded status flag, FPSCR_{FR}.
- n The value derived when the source operand, b, is rounded to an integer using the special rounding for Round-To-FP-Integer.
- P(x) The QNaN of operand x is propagated and placed in FRT[p].
- T(x) The value x is placed in FRT[p].
- TV The system floating-point enabled exception error handler is invoked for the invalid-operation exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode.
- U(x) The SNaN of operand x is converted to the corresponding QNaN and placed in FRT[p].
- W(x) The value x in the form of zero exponent or the source exponent is placed in FRT[p].

Figure 32. Actions: Round to FP Integer Without Inexact

1.6.5 DFP Conversion Instructions

The DFP conversion instructions consist of data-format conversion instructions and data-type conversion instructions. They are all X-form instructions and have the record bit provided.

1.6.5.1 DFP Data-Format Conversion Instructions

The data-format conversion instructions consist of *Convert To DFP64*, *Convert To DFP128*, *Round To DFP32*, and *Round To DFP64*. The following table summarizes the actions for these instructions.

Instruction	Actions when operand b in FRB[p] is			
	F	∞	QNaN	SNaN
Convert To DFP64	$T(b)^1$	$P(b)^{2,4}$	$P(b)^{2,4}$	$P(b)^{3,4}$
Convert To DFP128	$T(b)^1$	$T(dINF)$	$P(b)^{2,4}$	$V_{XSNAN}: U(b)^{2,4}$
Round To DFP32	$R(b)^1$	$P(b)^{2,5}$	$P(b)^{2,5}$	$P(b)^{3,5}$
Round To DFP64	$R(b)^1$	$T(dINF)$	$P(b)^{2,5}$	$V_{XSNAN}: U(b)^{2,5}$

Explanation:

1	The ideal exponent is the exponent of the source operand.
2	The BEC field is set to zero.
3	Bits in the BEC field are set to zeros, except for bit 0 which is set to 1.
4	The CC field is padded on the left with zeros.
5	Leftmost digits in the CC field are removed.
dINF	Default infinity.
F	All finite numbers, including zeros.
P(x)	The special symbol in operand x is propagated into FRT[p].
R(x)	The value x is rounded to the target-format precision; see Section 1.5.11
T(x)	The value x is placed in FRT[p].
U(x)	The SNaN of operand x is converted to the corresponding QNaN.
V_{XSNAN}	The Invalid-Operation Exception (VXSNAN) occurs. The result is produced only when the exception is disabled. See Section 1.5.10.1 for actions.

Figure 33. Actions: Data-Format Conversion Instructions

Programming Note

DFP does not provide the *BFP Load Floating-Point Single* and *Store Floating-Point Single* functions. To facilitate the program to simulate these two functions, conversion of an SNaN or infinity between DFP32 and DFP64 simply shortens or lengthens the CC field with the contents being reencoded. If the SNaN or infinity in the DFP32 format uses the preferred DPD encoding, then converting this operand to the DFP64 format and back to DFP32 will result in the original bit pattern.

DFP Convert To DFP64 X-form

dctdp FRT,FRB (Rc=0)
 dctdp. FRT,FRB (Rc=1)

59	FRT	//	FRB	258	Rc
0	6	11	16	21	31

The DFP operand in the source register FRB is converted to the DFP64 format and the converted result is placed into the target register FRT. The sign of the result is the same as the sign of the source operand. The ideal exponent is the exponent of the source operand.

The source operand is 32 bits; it is in the DFP32 format and is in bits 32-63 of FRB. Bits 0-31 of FRB are ignored. If the source operand is an SNaN, it is converted to an SNaN in the DFP64 format and does not cause an invalid-operation exception.

Special Registers Altered:

FPRF
 CR1 (if Rc=1)

DFP Convert To DFP128 X-form

dctqpq FRTp,FRB (Rc=0)
 dctqpq. FRTp,FRB (Rc=1)

63	FRTp	//	FRB	258	Rc
0	6	11	16	21	31

The DFP operand in the source register FRB is converted to the DFP128 format and the converted result is placed into the target register pair FRTp. The sign of the result is the same as the sign of the source operand. The ideal exponent is the exponent of the source operand.

The source operand is 64 bits and is in the DFP64 format. If the source operand is an SNaN, an invalid-operation exception is recognized. If the exception is disabled, the source SNaN is converted to the corresponding QNaN in the DFP128 format and the converted QNaN is returned.

Special Registers Altered:

FPRF FR FI (FR and FI are set to zero)
 FX
 VXSNAN
 CR1 (if Rc=1)

1.6.5.2 DFP Data-Type Conversion Instructions

The DFP data-type conversion instructions are used to convert data type between DFP and fixed.

The data-type conversion instructions consist of *Convert From Fixed* and *Convert To Fixed*.

DFP Convert From Fixed Quad X-form

dccfixq FRTp,FRB (Rc=0)
 dccfixq. FRTp,FRB (Rc=1)

63	FRTp	//	FRB	802	Rc
0	6	11	16	21	31

The 64-bit signed binary integer in the source register FRB is converted and rounded to a DFP128 value and placed into the target register FRTp. The sign of the result is the same as the sign of the source operand. The ideal exponent is zero.

If the source operand is a zero, then a plus zero with a zero exponent is returned.

The following table summarizes the actions for *Convert From Fixed*. The table does not include the setting of the FPSCR_{FPRF} field. The FPSCR_{FPRF} field is always set to the class and sign of the result.

Special Registers Altered:

FPRF
 CR1 (if Rc=1)

Actions when operand b is					
-0	+0	Fn	∞	QNaN	SNaN
--	T(+0)	Rf(b)	--	--	--
Explanation:					
--	Not applicable.				
Fn	Finite nonzero numbers (includes both sub-normal and normal numbers).				
QNaN	Quiet NaN.				
Rf(b)	The value b is converted to the precise DFP number, and then rounded to the target-format precision. See Section 1.5.11 on page 21.				
SNaN	Signaling NaN.				
T(x)	The value x is placed in FRTp.				

Figure 34. Actions: Convert From Fixed

DFP Convert To Fixed [Quad] X-form

dctfix FRT,FRB (Rc=0)
 dctfix. FRT,FRB (Rc=1)

59	FRT	//	FRB	290	Rc
0	6	11	16	21	31

dctfixq FRT,FRBp (Rc=0)
 dctfixq. FRT,FRBp (Rc=1)

63	FRT	//	FRBp	290	Rc
0	6	11	16	21	31

The DFP operand in the source register FRB[p] is rounded to an integer value and is placed into the target register FRT in the 64-bit signed binary integer format. The sign of the result is the same as the sign of the source operand, except when the source operand is a NaN or a zero.

The following table summarizes the actions for *Convert To Fixed*.

Special Registers Altered:

FR FI
 FX XX
 VXSNaN VXCVI
 CR1 (if Rc=1)

Operand b in FRB[p] is	q is	Is n not precise ($n \neq b$)	Inv.-Op. Except. Enabled	Inexact Except. Enabled	Is n Incremented ($ n > b $)	Actions *
$-\infty \leq b < MN$	$< MN$	-	No	-	-	$T(MN), FI \leftarrow 0, FR \leftarrow 0, VXCVI \leftarrow 1$
$-\infty \leq b < MN$	$< MN$	-	Yes	-	-	$VXCVI \leftarrow 1, TV$
$-\infty < b < MN$	$= MN$	-	-	No	-	$T(MN), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1$
$-\infty < b < MN$	$= MN$	-	-	Yes	-	$T(MN), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1, TX$
$MN \leq b < 0$	-	No	-	-	-	$T(n), FI \leftarrow 0, FR \leftarrow 0$
$MN \leq b < 0$	-	Yes	-	No	No	$T(n), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1$
$MN \leq b < 0$	-	Yes	-	No	Yes	$T(n), FI \leftarrow 1, FR \leftarrow 1, XX \leftarrow 1$
$MN \leq b < 0$	-	Yes	-	Yes	No	$T(n), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1, TX$
$MN \leq b < 0$	-	Yes	-	Yes	Yes	$T(n), FI \leftarrow 1, FR \leftarrow 1, XX \leftarrow 1, TX$
± 0	-	No	-	-	-	$T(0), FI \leftarrow 0, FR \leftarrow 0$
$0 < b \leq MP$	-	No	-	-	-	$T(n), FI \leftarrow 0, FR \leftarrow 0$
$0 < b \leq MP$	-	Yes	-	No	No	$T(n), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1$
$0 < b \leq MP$	-	Yes	-	No	Yes	$T(n), FI \leftarrow 1, FR \leftarrow 1, XX \leftarrow 1$
$0 < b \leq MP$	-	Yes	-	Yes	No	$T(n), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1, TX$
$0 < b \leq MP$	-	Yes	-	Yes	Yes	$T(n), FI \leftarrow 1, FR \leftarrow 1, XX \leftarrow 1, TX$
$MP < b < +\infty$	$= MP$	-	-	No	-	$T(MP), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1$
$MP < b < +\infty$	$= MP$	-	-	Yes	-	$T(MP), FI \leftarrow 1, FR \leftarrow 0, XX \leftarrow 1, TX$
$MP < b \leq +\infty$	$> MP$	-	No	-	-	$T(MP), FI \leftarrow 0, FR \leftarrow 0, VXCVI \leftarrow 1$
$MP < b \leq +\infty$	$> MP$	-	Yes	-	-	$VXCVI \leftarrow 1, TV$
QNaN	-	-	No	-	-	$T(MN), FI \leftarrow 0, FR \leftarrow 0, VXCVI \leftarrow 1$
QNaN	-	-	Yes	-	-	$VXCVI \leftarrow 1, TV$
SNaN	-	-	No	-	-	$T(MN), FI \leftarrow 0, FR \leftarrow 0, VXCVI \leftarrow 1, VXSNaN \leftarrow 1$
SNaN	-	-	Yes	-	-	$VXCVI \leftarrow 1, VXSNaN \leftarrow 1, TV$

Explanation:

- * Setting of XX, VXCVI, and VXSNaN is part of the corresponding exception actions. Also, when an invalid-operation exception occurs, setting of FI and FR bits is part of the exception actions. (See the sections, "Inexact Exception" and "Invalid Operation Exception" for more details.)
- The actions do not depend on this condition.
- FI Floating-Point-Fraction-Inexact status flag, $FPSCR_{FI}$.
- FR Floating-Point-Fraction-Rounded status flag, $FPSCR_{FR}$.
- MN Maximum negative number representable by the 64-bit binary integer format
- MP Maximum positive number representable by the 64-bit binary integer format.
- n The value q converted to a fixed-point result.
- q The value derived when the source value b is rounded to an integer using the specified rounding mode
- $T(x)$ The value x is placed in $FRT[p]$.
- TV The system floating-point enabled exception error handler is invoked for the invalid-operation exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode.
- TX The system floating-point enabled exception error handler is invoked for the inexact exception if the FE0 and FE1 bits in the machine-state register are set to any mode other than the ignore-exception mode.
- VXCVI The $FPSCR_{VXCVI}$ invalid operation exception status bit.
- VXSNaN The $FPSCR_{VXSNaN}$ invalid operation exception status bit.
- XX Floating-Point-Inexact-Exception status flag, $FPSCR_{XX}$.

Figure 35. Actions: Convert To Fixed

1.6.6 DFP Format Instructions

The DFP format instructions are used to compose or decompose a DFP operand. A source operand of SNaN does not cause an invalid-operation exception. All format instructions have the record bit provided.

The format instructions consist of *Decode DPD To BCD*, *Encode BCD To DPD*, *Extract Biased Exponent*, *Insert Biased Exponent*, *Shift Coefficient Left Immediate*, and *Shift Coefficient Right Immediate*.

DFP Decode DPD To BCD [Quad] X-form

ddedpd SP,FRT,FRB (Rc=0)
 ddedpd. SP,FRT,FRB (Rc=1)

59	FRT	SP	///	FRB	322	Rc
0	6	11	13	16	21	31

ddedpdq SP,FRTp,FRBp (Rc=0)
 ddedpdq. SP,FRTp,FRBp (Rc=1)

63	FRTp	SP	///	FRBp	322	Rc
0	6	11	13	16	21	31

The **ddedpd[q]** instruction converts a portion of the coefficient of the DFP operand in FRB[p] to a signed or unsigned BCD number depending on the SP field. For infinity and NaN, the coefficient is considered to be the contents in the CC field padded on the left by a zero digit.

SP₀ = 0 (unsigned conversion)

The rightmost 16 digits of the coefficient (32 digits for **ddedpdq**) is converted to an unsigned BCD number and the result is placed into FRT[p].

SP₀ = 1 (signed conversion)

The rightmost 15 digits of the coefficient (31 digits for **ddedpdq**) is converted to a signed BCD number with the same sign as the DFP operand, and the result is placed into FRT[p]. If the DFP operand is negative, the sign is encoded as 0b1101. If the DFP operand is positive, SP1 indicates which preferred plus sign encoding is used. If SP1 = 0, the plus sign is encoded as 0b1100 (the option-1 preferred sign code), otherwise the plus sign is encoded as 0b1111 (the option-2 preferred sign code).

Special Registers Altered:

CR1 (if Rc=1)

DFP Encode BCD To DPD [Quad] X-form

denbcd S,FRT,FRB (Rc=0)
 denbcd. S,FRT,FRB (Rc=1)

59	FRT	S	////	FRB	834	Rc
0	6	11	12	16	21	31

denbcdq S,FRTp,FRBp (Rc=0)
 denbcdq. S,FRTp,FRBp (Rc=1)

63	FRTp	S	////	FRBp	834	Rc
0	6	11	12	16	21	31

The **denbcd[q]** instruction converts the signed or unsigned BCD operand, depending on the S field, in FRB[p] into a DFP number. The ideal exponent is zero

S = 0 (unsigned BCD operand)

The unsigned BCD operand in register FRB[p] is converted to a positive DFP number of the same magnitude and the result is placed into register FRT[p].

S = 1 (signed BCD operand)

The signed BCD operand in register FRB[p] is converted to the corresponding DFP number and the result is placed into register FRT[p].

If an invalid BCD digit or sign code is detected in the source operand, an invalid-operation exception (VXCVI) occurs.

FPSCR_{FPRF} is set to the class and sign of the result, except for Invalid Operation Exception when FPSCR_{VE}=1.

Special Registers Altered:

FPRF FR FI (FR and FI are set to zero)
 FX
 VXCVI
 CR1 (if Rc=1)

Operand a in FRA[p] specifies	Actions for Insert Biased Exponent when operand b in FRB[p] specifies			
	F	∞	QNaN	SNaN
F	N, Rb	Z, Rb	Z, Rb	Z, Rb
∞	I, Rb	I, Rb	I, Rb	I, Rb
QNaN	Q, Rb	Q, Rb	Q, Rb	Q, Rb
SNaN	S, Rb	S, Rb	S, Rb	S, Rb

Explanation:

F	All finite numbers, including zeros
I	The combination field in FRT[p] is set to indicate an infinity and the biased-exponent continuation (BEC) field is set to zero.
N	The combination and the BEC fields in FRT[p] are set to the specified biased exponent in FRA and the leftmost coefficient digit in FRB[p].
Q	The combination field in FRT[p] is set to indicate a NaN and the BEC field is set to zero.
S	The combination field in FRT[p] is set to indicate a NaN, and the leftmost bit in the BEC field is set to one and the remaining bits in BEC set to zero.
Z	The combination and the BEC fields in FRT[p] are set to the specified biased exponent in FRA and a leftmost coefficient digit of zero.
Rb	The contents of the coefficient-continuation field in FRB[p] are reencoded using preferred DPD codes and the reencoded result is placed in the same field in FRT[p]. The sign bit of FRB[p] is copied into the sign bit in FRT[p].

Figure 36. Actions: Insert Biased Exponent

1.6.7 DFP Instruction Summary

Mnemonic	Full Name	FORM	Operands	SNaN Vs G	Encoding*	FPRF		FP Exception				FR/FI	IE	Rc
						C	FPC	V	Z	O	U			
dadd	DFP Add	X	FRT, FRA, FRB	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
daddq	DFP Add Quad	X	FRTp, FRAp, FRBp	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
dsub	DFP Subtract	X	FRT, FRA, FRB	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
dsubq	DFP Subtract Quad	X	FRTp, FRAp, FRBp	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
dmul	DFP Multiply	X	FRT, FRA, FRB	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
dmulq	DFP Multiply Quad	X	FRTp, FRAp, FRBp	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
ddiv	DFP Divide	X	FRT, FRA, FRB	Y N	RE	Y	Y	V	Z	O	U	X	Y	Y
ddivq	DFP Divide Quad	X	FRTp, FRAp, FRBp	Y N	RE	Y	Y	V	Z	O	U	X	Y	Y
dcmpo	DFP Compare Ordered	X	BF, FRA, FRB	Y -	-	N	Y	V				-	-	N
dcmpoq	DFP Compare Ordered Quad	X	BF, FRAp, FRBp	Y -	-	N	Y	V				-	-	N
dcmpu	DFP Compare Unordered	X	BF, FRA, FRB	Y -	-	N	Y	V				-	-	N
dcmpuq	DFP Compare Unordered Quad	X	BF, FRAp, FRBp	Y -	-	N	Y	V				-	-	N
dtstdc	DFP Test Data Class	Z	BF, FRA, DCM	N -	-	N	Y ¹					-	-	N
dtstdcq	DFP Test Data Class Quad	Z	BF, FRAp, DCM	N -	-	N	Y ¹					-	-	N
dtstdg	DFP Test Data Group	Z	BF, FRA, DGM	N -	-	N	Y ¹					-	-	N
dtstdgq	DFP Test Data Group Quad	Z	BF, FRAp, DGM	N -	-	N	Y ¹					-	-	N
dtstex	DFP Test Exponent	X	BF, FRA, FRB	N -	-	N	Y					-	-	N
dtstexq	DFP Test Exponent Quad	X	BF, FRAp, FRBp	N -	-	N	Y					-	-	N
dtstsf	DFP Test Significance	X	BF, FRA(FIX), FRB	N -	-	N	Y					-	-	N
dtstsfq	DFP Test Significance Quad	X	BF, FRA(FIX), FRBp	N -	-	N	Y					-	-	N
dquai	DFP Quantize Immediate	Z	TE, FRT, FRB, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
dquaiq	DFP Quantize Immediate Quad	Z	TE, FRTp, FRBp, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
dqua	DFP Quantize	Z	FRT, FRA, FRB, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
dquaq	DFP Quantize Quad	Z	FRTp, FRAp, FRBp, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
drnd	DFP Reround	Z	FRT, FRA(FIX), FRB, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
drndq	DFP Reround Quad	Z	FRTp, FRA(FIX), FRBp, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
drintx	DFP Round To FP Integer With Inexact	Z	R, FRT, FRB, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
drintxq	DFP Round To FP Integer With Inexact Quad	Z	R, FRTp, FRBp, RMC	Y N	RE	Y	Y	V		X	Y	Y	Y	Y
drintn	DFP Round To FP Integer Without Inexact	Z	R, FRT, FRB, RMC	Y N	RE	Y	Y	V				Y [#]	Y	Y
drintnq	DFP Round To FP Integer Without Inexact Quad	Z	R, FRTp, FRBp, RMC	Y N	RE	Y	Y	V				Y [#]	Y	Y
dctdp	DFP Convert To DFP64	X	FRT, FRB (DFP32)	N Y	RE	Y	Y ²					-	Y	Y
dctpq	DFP Convert To DFP128	X	FRTp, FRB	Y N	RE	Y	Y	V				Y [#]	Y	Y
drsp	DFP Round To DFP32	X	FRT (DFP32), FRB	N Y	RE	Y	Y ²		O	U	X	Y	Y	Y
drdpq	DFP Round To DFP64	X	FRTp, FRBp	Y N	RE	Y	Y	V	O	U	X	Y	Y	Y
dcffixq	DFP Convert From Fixed Quad	X	FRTp, FRB (FIX)	- N	RE	Y	Y					-	Y	Y
dctfix	DFP Convert To Fixed	X	FRT (FIX), FRB	Y N	-	N	N	V		X	Y	-	Y	Y
dctfixq	DFP Convert To Fixed Quad	X	FRT (FIX), FRBp	Y N	-	N	N	V		X	Y	-	Y	Y
ddedpd	DFP Decode DPD To BCD	X	SP, FRT(BCD), FRB	N -	-	N	N					-	-	Y
ddedpdq	DFP Decode DPD To BCD Quad	X	SP, FRTp(BCD), FRBp	N -	-	N	N					-	-	Y

Preliminary - Subject to change

Mnemonic	Full Name	FORM	Operands	SNaN Vs G	Encoding*	FPRF		FP Exception V Z O U X	FR/IE	Rc	
						C	EPCC				
denbcd	DFP Encode BCD To DPD	X	S, FRT, FRB (BCD)	- N	RE	Y	Y	V	Y#	Y	Y
denbcdq	DFP Encode BCD To DPD Quad	X	S, FRTp, FRBp (BCD)	- N	RE	Y	Y	V	Y#	Y	Y
dxex	DFP Extract Biased Exponent	X	FRT (FIX), FRB	N N	-	N	N		-	-	Y
dxexq	DFP Extract Biased Exponent Quad	X	FRT (FIX), FRBp	N N	-	N	N		-	-	Y
diex	DFP Insert Biased Exponent	X	FRT, FRA(FIX), FRB	N Y	RE	N	N		-	Y	Y
diexq	DFP Insert Biased Exponent Quad	X	FRTp, FRA(FIX), FRBp	N Y	RE	N	N		-	Y	Y
dscli	DFP Shift Coefficient Left Immediate	Z	FRT,FRA,SH	N Y	RE	N	N		-	-	Y
dscliq	DFP Shift Coefficient Left Quad Immediate	Z	FRTp,FRAp,SH	N Y	RE	N	N		-	-	Y
dscri	DFP Shift Coefficient Right Immediate	Z	FRT,FRA,SH	N Y	RE	N	N		-	-	Y
dscriq	DFP Shift Coefficient Right Quad Immediate	Z	FRTp,FRAp,SH	N Y	RE	N	N		-	-	Y
Explanation:											
#	FI and FR are set to zeros for these instructions.										
*	See next table for more details.										
-	Not applicable.										
1	A unique definition of the FPSCR _{FPCC} field is provided for the instruction.										
2	These are the only instructions that may generate an SNaN and also set the FPSCR _{FPRF} field. Since the BFP FPSCR _{FPRF} field does not include a code for SNaN, these instructions cause the need for redefining the FPSCR _{FPRF} field for DFP.										
DCM	A 6-bit immediate operand specifying the data-class mask.										
DGM	A 6-bit immediate operand specifying the data-group mask.										
G	An SNaN can be generated as the target operand.										
IE	An ideal exponent is defined for the instruction.										
FI	Setting of the FPSCR _{FI} flag.										
FR	Setting of the FPSCR _{FR} flag.										
N	No.										
O	An overflow exception may be recognized.										
Rc	The record bit, Rc, is provided to record FPSCR _{0:3} in CR field 1.										
RE	The CC field is reencoded using preferred DPD encodngs.The preferred DPD encoding are also used for propagated NaNs, or converted NaNs and infinities.										
RMC	A 2-bit immediate operand specifying the rounding-mode control.										
S	An one-bit immediate operand specifying if the operation is signed or unsigned.										
SP	A two-bit immediate operand: one bit specifies if the operation is signed or unsigned and, for signed operations, another bit specifies which preferred plus sign code is generated.										
U	An underflow exception may be recognized.										
V	An invalid-operation exception may be recognized.										
Vs	An input operand of SNaN causes an invalid-operation exception.										
X	An inexact exception may be recognized.										
Y	Yes.										
Z	A zero-divide exception may be recognized.										

Figure 37. Decimal Floating-Point Instructions Summary

