

RH850G3MH

User's Manual: Software

Renesas microcontroller

All information contained in these materials, including products and product specifications, represents information on the product at the time of publication and is subject to change by Renesas Electronics Corp. without notice. Please review the latest information published by Renesas Electronics Corp. through various means, including the Renesas Electronics Corp. website (http://www.renesas.com).

Renesas Electronics www.renesas.com

Rev.1.00 Mar, 2015

Notice

- 1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
- 2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
- 3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
- 4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
- 5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.

"Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.

"High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anticrime systems; and safety equipment etc.

Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.

- 6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
- 7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
- 8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
- 9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
- 10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
- 11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
- 12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.
- (Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majorityowned subsidiaries.
- (Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

NOTES FOR CMOS DEVICES

- (1) VOLTAGE APPLICATION WAVEFORM AT INPUT PIN: Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between VIL (MAX) and VIH (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between VIL (MAX) and VIH (MIN).
- (2) HANDLING OF UNUSED INPUT PINS: Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to VDD or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.
- (3) PRECAUTION AGAINST ESD: A strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.
- (4) STATUS BEFORE INITIALIZATION: Power-on does not necessarily define the initial status of a MOS device. Immediately after the power source is turned ON, devices with reset functions have not yet been initialized. Hence, power-on does not guarantee output pin levels, I/O settings or contents of registers. A device is not initialized until the reset signal is received. A reset operation must be executed immediately after power-on for devices with reset functions.
- (5) POWER ON/OFF SEQUENCE: In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current. The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.
- (6) INPUT OF SIGNAL DURING POWER OFF STATE : Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

How to Use This Manual

| Target and Readers | This manual is intended for users who wish to understand the RH850G3M software and design application systems using these products. | | | |
|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|--|--|
| Conventions | Data significance: | Higher digits on the left and lower digits on the right | | |
| | Active low representation: | xxx (overscore over pin or signal name) | | |
| | Memory map address: | Higher addresses on the top and lower addresses on the bottom | | |
| | Note: | Footnote for item marked with Note in the text | | |
| | Caution: | Information requiring particular attention | | |
| | Remark: | Supplementary information | | |
| | Numeric representation: | Binary xxxx or xxxx _B | | |
| | | Decimal xxxx | | |
| | | Hexadecimal xxxx _H | | |
| | Prefix indicating power of 2 (address space, memory capacity): | | | |
| K (kilo): 2 ¹⁰ = 1,024 | | K (kilo): 2 ¹⁰ = 1,024 | | |

M (mega): 2²⁰ = 1,024²

G (giga): 2³⁰ = 1,024³

All trademarks and registered trademarks are the property of their respective owners.

Table of Contents

| Sectio | n 1 | OVERVIEW | 8 |
|--------|-----------------|----------------------------------------------------------------------|----------|
| 1.1 | Fea | atures of the RH850G3MH | 8 |
| 1.2 | Ch | anges from the RH850G3M | 9 |
| Soctio | n 2 | | 10 |
| Sectio | // / Z | | 10 |
| 2.1 | CP | U Operating Modes | 10 |
| | 2.1.1 | Definition of CPU Operating Modes | 10 |
| | 2.1.2 | CPU Operating Mode Transition | 11 |
| | 2.1.3 | CPU Operating Modes and Privileges | 12 |
| 2.2 | Ins | truction Execution | 14 |
| 2.3 | Exc | ceptions and Interrupts | 16 |
| | 2.3.1 | Exception Level | 16 |
| 2.4 | Co | Drocessors | 17 |
| | 2.4.1 | Coprocessor Use Permissions | 17 |
| | 2.4.2 | Correspondences between Coprocessor Use Permissions and Coprocessors | 17 |
| | 2.4.3 | Coprocessor Unusable Exceptions | 17 |
| | 2.4.4 | System Registers | 17 |
| 2.5 | Re | gisters | 18 |
| | 2.5.1 | Program Registers | 18 |
| | 2.5.2 | System Registers | 18 |
| | 2.5.3 | Register Updating | 18 |
| | 2.5.4 | Accessing Undefined Registers | 20 |
| 2.6 | Da | a Types | 21 |
| | 2.6.1 | Data formats | 21 |
| | 2.6.2 | Data Representation | 23 |
| | 2.6.3 | Data Alignment | 24 |
| 2.7 | Ad | dress Space | 26 |
| | 2.7.1 | Memory Map | 27 |
| | 2.7.2 | Instruction Addressing | 28 |
| | 2.7.3 | Data Addressing | 30 |
| 2.8 | Aco | quiring the CPU Number | 35 |
| 2.9 | Sys | stem Protection Identifier | 35 |
| Sectio | n 3 | REGISTER SET | 36 |
| 3.1 | Pro | gram Registers | 36 |
| | 3.1.1 | General-Purpose Registers | 37 |
| | 3.1.2 | PC — Program Counter | 38 |
| 3.2 | Ba | sic Svstem Registers | 39 |
| 3.3 | Inte | errupt Function Registers | 58 |
| 0.0 | 3.3.1 | Interrupt Function System Registers | 58 |
| 2 / | FD | I Function Registers | 62 |
| 5.4 | 3/1 | Floating-Doint Registers | 20 |
| | 3.4.1 3.⁄1.2 | Floating-Point Function System Registers | 02 62 |
| | J.4.Z | | 02 |

| 3.5 | MF | PU Function Registers | 69 |
|--------|---------------------|-----------------------------------------------------------------------|-----|
| | 3.5.1 | MPU Function System Registers | 69 |
| 3.6 | Ca | che Operation Function Registers | 77 |
| | 3.6.1 | Cache Control Function System Registers | 77 |
| 3.7 | Da | ta Buffer Operation Registers | 83 |
| | 3.7.1 | Data Buffer Control System Registers | 83 |
| | | | |
| Sectio | on 4 | EXCEPTIONS AND INTERRUPTS | |
| 4.1 | Ou | tline of Exceptions | |
| | 4.1.1 | Exception Cause List | |
| | 4.1.2 | Overview of Exception Causes | 87 |
| | 4.1.3 | Types of Exceptions | 88 |
| | 4.1.4 | Exception Acknowledgment Conditions and Priority Order | 89 |
| | 4.1.5 | Interrupt Exception Priority and Priority Masking | |
| | 4.1.6 | Return and Restoration | 91 |
| | 4.1.7 | Context Saving | |
| 4.2 | Ор | eration When Acknowledging an Exception | |
| | 4.2.1 | Special Operations | 95 |
| 4.3 | Re | turn from Exception Handling | |
| 4.4 | Ex | ception Handler Address | |
| | 4.4.1 | Resets, Exceptions, and Interrupts | |
| | 4.4.2 | System Calls | 103 |
| | 4.4.3 | Models for Application | 104 |
| Sactio | n 5 | | 106 |
| 56010 | | | 100 |
| 5.1 | Me | mory Protection Unit (MPU) | 106 |
| | 5.1.1 | Features | |
| | 5.1.2 | Protection Area Settings | |
| | 5.1.3 | Caution Points for Protection Area Setup | |
| | 5.1.4 | Access Control | |
| | 5.1.5 | Violations and Exceptions | |
| 5.0 | 0.1.0 | | |
| 5.2 | Ca | | |
| | 5.2.1 | Cache Operation Registers | |
| | 5.2.2 | Change Cache Use Mode | |
| | 5.2.3 | Cache Operations using CACHE Instruction | |
| | 5.2.4 | Cache Index Specification Method | 114 |
| | 526 | Execution Privilege of the CACHE/PREE Instruction | |
| | 527 | Memory Protection for CACHE and PREF Instructions | 116 |
| 53 | <u>0.2.</u> , Мі | | |
| 0.0 | 521 | Shared Data that does not Require Mutual Evolution Processing | |
| | 532 | Performing Mutual Exclusion by Using the LDL W and STC W Instructions | |
| | 533 | Performing Mutual Exclusion by Using the SET1 Instruction | |
| | 0.0.0 | | |

| | 5.3.4 | Performing Mutual Exclusion by Using the CAXI Instruction | 121 |
|------------|--------|-----------------------------------------------------------|-----|
| Sectio | on 6 C | COPROCESSOR | 122 |
| 6.1 | Float | ing-Point Operation | 122 |
| | 6.1.1 | Configuration of Floating-Point Operation Function | 122 |
| | 6.1.2 | Data Types | 123 |
| | 6.1.3 | Register Set | 126 |
| | 6.1.4 | Floating-Point Instructions | 126 |
| | 6.1.5 | Floating-Point Operation Exceptions | 127 |
| | 6.1.6 | Exception Details | 130 |
| | 6.1.7 | Saving and Returning Status | 134 |
| | 6.1.8 | Flushing Subnormal Numbers | 135 |
| | 6.1.9 | Flush to Nearest | 137 |
| Sectio | on 7 l | NSTRUCTION | 138 |
| 71 | Opco | odes and Instruction Formats | 138 |
| | 711 | | 138 |
| | 712 | Coprocessor Instructions | 143 |
| | 713 | Reserved Instructions | 143 |
| 72 | Basic | Instructions | 144 |
| 1.2 | 721 | Overview of Basic Instructions | 144 |
| | 7.2.2 | Basic Instruction Set | 149 |
| 73 | Cach | | 300 |
| 1.0 | 731 | Overview of Cache Instructions | 300 |
| | 732 | Cache Instruction Set | 300 |
| 7 / | T.J.Z | | 205 |
| 7.4 | FIUAL | | 305 |
| | 7.4.1 | Instruction formats | 305 |
| | 7.4.2 | Overview of Floating-Point Instructions | 306 |
| | 7.4.3 | Conditions for Comparison Instructions | 309 |
| | 7.4.4 | Floating-Point Instruction Set | 311 |
| Sectio | n 8 F | RESET | 407 |
| 8.1 | Statu | is of Registers After Reset | 407 |
| APPE | NDIX A | HAZARD RESOLUTION PROCEDURE FOR SYSTEM REGISTERS | 408 |
| APPE | NDIX B | NUMBER OF INSTRUCTION EXECTUION CLOCKS | 409 |
| R 1 | Num | bers of Clock Cycles for Execution | 409 |
| ן.ם פיס | Num | ber of G3MH Instruction Execution Clocks | /10 |
| D.2 | num | | +10 |
| APPE | NDIX C | REGISTER INDEX | 419 |
| APPE | NDIX D | INSTRUCTION INDEX | 420 |

Section 1 OVERVIEW

1.1 Features of the RH850G3MH

The RH850G3MH features compatibility with the instruction set for all 32-bit RISC microcontrollers of the RH850G3M Series, but has even better performance.

Table 1.1 shows the features of the RH850G3MH.

Table 1.1 Features of the RH850G3MH

| Item | Features |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CPU | High performance 32-bit architecture for embedded control |
| | 32-bit internal data bus |
| | Thirty-two 32-bit general-purpose registers |
| | RISC type instruction set (compatible with RH850G3M) Long/short type load/store instructions Three-operand instructions Instruction set based on C |
| | CPU operating modes User mode and supervisor mode |
| | Address space: 4-Gbyte linear space for both data and instructions |
| Coprocessor | A floating point operation coprocessor (FPU) can be installed. Supports single precision (32-bit) and double precision (64-bit) Supports IEEE754-compliant data types and exceptions Rounding modes : Nearest, 0 direction, +∞ direction, and -∞ direction Handling on non-normalized numbers: These are truncated to 0, or an exception is reported because such numbers do not comply with IEEE754. |
| Exceptions/interrupts | 16-level interrupt priority that can be specified for each channel Vector selection method that can be selected according to performance requirements and the amount of consumed memory Direct branch method exception vector (direct vector method) Address-table-referencing indirect branch method exception vector (table reference method) Support for high-speed context backup and restoration processing on interrupt by |
| | using dedicated instructions (PUSHSP, POPSP) |
| Memory management | A memory protection unit (MPU) can be installed. |
| Caches | An instruction cache can be installed. |



1.2 Changes from the RH850G3M

| Item | Changes | | |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| CPU | The specifications of system registers listed below were changed. The specifications of MEI and MCTL were changed (see Section 3.2, Basic System Registers). | | |
| | The function of the SYNCE instruction was changed (see Section 7.2.2, Basic Instruction Set). | | |
| | The hazard resolution procedure does not proceed after updating of SCCFG as it was found to be unnecessary in that case (see APPENDIX A, HAZARD RESOLUTION PROCEDURE FOR SYSTEM REGISTERS). | | |
| Exceptions/interrupts | The specifications of system registers were changed and a system register was deleted as stated below. | | |
| | FPIR was deleted, and the specifications of ISPR, PMR, and ICSR were changed (see Section 3.3.1, Interrupt Function System Registers). | | |
| | Changes were made to the exceptions for floating-point operations. The FPP exception and FPI exception were abolished. An FPINT exception was added. | | |
| Coprocessor | The specification of a system register was changed and a system register was deleted as stated below. The specification of FPSR was changed and FPEC was deleted (see Section 3.4.2, Floating-Point Function System Registers). | | |
| | The hazard resolution procedure does not proceed after updating of any of the FPU registers (FPSR, FPEPC, FPST, FPCC, or FPCFG) as it was found to be unnecessary in these cases (see APPENDIX A, HAZARD RESOLUTION PROCEDURE FOR SYSTEM REGISTERS). | | |
| Memory management | A specification of the MPU was changed. The operation in response to memory access spanning contiguous areas to which access is enabled by the MPU was changed (see Section 5.1.3, Caution Points for Protection Area Setup). | | |
| Cache | The specifications of system registers listed below were changed. The specifications of ICCTRL, ICTAGL, ICTAGH, and ICCFG were changed (see Section 3.6.1, Cache Control Function System Registers). | | |
| | The wait for the completion of clearing of the instruction cache by the ICCTRL.ICHCLR bit is not necessary (see APPENDIX A, HAZARD RESOLUTION PROCEDURE FOR SYSTEM REGISTERS). | | |

Table 1.2Changes from the RH850G3M



Section 2 PROCESSOR MODEL

This CPU defines a processor model that has basic operation functions, registers, and an exception management function.

This section describes the unique features of the processor model of this CPU.

2.1 CPU Operating Modes

This CPU has defines two operating statuses of the supervisor mode (SV) and the user mode (UM). Whether the system is in supervisor mode or user mode is indicated by the UM bit in the PSW register.

- Supervisor mode (PSW.UM = 0) : All hardware functions can be managed or used.
- User mode (PSW.UM = 1) : The usable hardware functions are restricted.

2.1.1 Definition of CPU Operating Modes

(1) Supervisor mode (SV)

All hardware functions can be managed or used in this mode. The system always starts up in supervisor mode after the end of reset processing.

(2) User mode (UM)

This operating mode makes up a pair with the supervisor mode. In user mode, address spaces to which access is permitted by the supervisor and the system registers defined as user resources can be used. Supervisor-privileged instructions cannot be executed and result in exceptions if they are.

Restriction in user mode (PSW.UM = 1)

• Privileged instruction violations due to SV-privileged-instruction operating restrictions (\rightarrow PIE exceptions) For details about privileged-instruction operating restrictions, see Section 2.1.3, CPU Operating Modes and Privileges



2.1.2 CPU Operating Mode Transition

The CPU operating mode changes due to three events.

(1) Change due to acknowledging an exception

When an exception is acknowledged, the CPU operating mode changes to the mode specified for the exception.

(2) Change due to a return instruction

When a return instruction is executed, the PSW value is restored according to the value of the corresponding bit backed up to EIPSW and FEPSW.

(3) Change due to a system register instruction

The CPU operating mode changes when an LDSR instruction is used to directly overwrite the PSW operating mode bits.

CAUTIONS

- 1. In supervisor mode, the LDSR instruction can be used to directly change the value of the PSW.UM bit, but system-register-related hazards are defined in the hardware specifications.For the change of this bit, it is recommended to use a return instruction to avoid PSW-register-related hazards.
- 2. In user mode, the CPU operating mode cannot be changed because the higher 31 to 5 bits of the PSW register cannot be overwritten. The CPU operating mode might be changed in supervisor mode, but system register access-related hazards are defined in the hardware specifications. For the change of this bit, it is recommended to use a return instruction to avoid PSW-register-related hazards.



2.1.3 CPU Operating Modes and Privileges

In this CPU, the usable functions can be restricted according to usage permission settings for specific resources and the CPU operating mode. Specification instructions (including instructions that update specific system registers) can only be executed in the defined operating mode. The permissions necessary to execute these specification instructions are called "privileges" below. In operating modes that do not have privileges, these instructions are not executed and exceptions occur.

This CPU defines the following two types of privileges (and usage permission).

- Supervisor (SV) privilege : Important system resources operation, fatal error processing, privilege necessary for user-mode program execution management
- Coprocessor use permissions : Permissions necessary to use a coprocessor



Figure 2.1 CPU Operating Modes and Privileges



(1) Supervisor privilege (SV privilege)

The privilege necessary to perform the operation for important system resources, fatal error processing, and user-mode program execution management is called the supervisor privilege (SV privilege). This privilege is available in supervisor mode. The SV privilege is generally necessary to execute instructions used to perform the operation for important system resources, and these instructions are sometimes called SV privileged instructions.

(2) Coprocessor use permissions

Regardless of the CPU operating mode, it is possible to separately specify whether coprocessors can be used.

The CU2 to CU0 bits in the PSW register are used in supervisor mode to specify whether coprocessors can be used by each program. If the CU bits are not set to 1, a coprocessor unusable exception occurs when the corresponding coprocessor instruction is executed or the system register is accessed.

If no coprocessor is installed, it is not possible to set the corresponding CU bits to 1. The setting of the CU2 to CU0 bits is valid regardless of the CPU operating mode, and, if the supervisor accesses coprocessor system registers, it is necessary to set the CU2 to CU0 bits to enable coprocessor use.

(3) Operation when there is a privilege violation

When an attempt is made to execute a privileged instruction by someone who does not have the required privilege, a PIE exception or UCPOP exception occurs. **Table 2.1** shows the relationships between the operating mode, usage permission status, and whether instructions can be executed.

| | PSW | | | | |
|-------------------------------------------------------|-----|-----|-----|-----|-------------------------------|
| | UM | CU2 | CU1 | CU0 | Whether Operation is Possible |
| SV privileged instruction | 0 | _ | _ | _ | Possible |
| | 1 | _ | _ | _ | Not possible/PIE exception |
| Coprocessor instruction 1*1 | _ | _ | _ | 1 | Possible |
| (PSW.CU0 bit) | _ | _ | _ | 0 | Not possible/UCPOP exception |
| Coprocessor instruction 2*1 | _ | _ | 1 | _ | Possible |
| (PSW.CU1 bit) | _ | _ | 0 | _ | Not possible/UCPOP exception |
| Coprocessor instruction 3 ^{*1} | _ | 1 | _ | _ | Possible |
| (PSW.CU2 bit) | _ | 0 | _ | _ | Not possible/UCPOP exception |
| Instructions other than the above (user instructions) | _ | _ | _ | _ | Possible |

Table 2.1Operation When There is a Privilege Violation

Note 1. This includes the LDSR/STSR instruction for the coprocessor system register. **Note:** —: 0 or 1



2.2 Instruction Execution

The instruction execution flow of this CPU is shown below.



Figure 2.2 Instruction Execution Flow



If terminating exceptions can be acknowledged or if the execution privilege of the instruction is not satisfied, an exception occurs before the instruction is executed. If a resumable exception occurs during the execution of an instruction, the exception is acknowledged during execution of the instruction. In these cases, the result of instruction execution is not reflected in the registers or memory, and the CPU state before the instruction was executed is retained^{*1}.

For a pending exception such as a software exception, the exception is acknowledged after the result of instruction execution has been reflected.

Note 1. The following instructions might cause intermediate results to be reflected in the memory. PREPARE, DISPOSE, PUSHSP, POPSP



2.3 Exceptions and Interrupts

Exceptions and interrupts are exceptional events that cause the program under execution to branch to another program. Exceptions and interrupts are triggered by various sources, including interrupts from peripherals and program abnormalities.

For details, see Section 4, EXCEPTIONS AND INTERRUPTS.

2.3.1 Exception Level

In this CPU, if an exception with a high degree of urgency occurs while another exception is being processed, the urgent exception will be processed by priority. To make it possible to return to the interrupted exception handling after acknowledging the urgent exception, even if the context had not been saved to the memory, exception causes are managed in the following two hierarchical levels.

- EI level exception
- FE level exception

EI level exceptions are used for processing such as regular user processing, interrupt servicing, and OS processing. FE level exceptions are used to enable interrupts with a high degree of urgency for the system or exceptions from the memory management function that might occur during OS processing to be acknowledged even while an EI level exception is being processed.



2.4 Coprocessors

In this CPU, single-precision and double-precision FPU expansion functions are incorporated.

2.4.1 Coprocessor Use Permissions

To execute a coprocessor instruction or defined opcode processing, permission to use the corresponding coprocessor instruction is necessary. Coprocessor use permissions are specified by the PSW.CU2 to PSW.CU0 bits, and, if an attempt is made to execute an instruction for which the corresponding coprocessor use permission is cleared to 0, a coprocessor unusable exception (UCPOP) occurs.

2.4.2 Correspondences between Coprocessor Use Permissions and Coprocessors

This CPU defines coprocessor use permissions to control the availability of the coprocessor for each program during CPU operation. There are three coprocessor use permissions (CU0 to CU2), and their correspondences with the coprocessors are shown in the following table.

 Table 2.2
 Correspondences Between Coprocessor Use Permissions and Coprocessors

| Coprocessor Use Permission | Coprocessor Function | Exception Cause Code |
|----------------------------|-----------------------------------------|----------------------|
| CU0 | Single-precision FPU expansion function | 80 _H |
| | Double-precision FPU expansion function | - |
| CU1 | Reserved | 81 _H |
| CU2 | Reserved | 82 _H |

2.4.3 Coprocessor Unusable Exceptions

A coprocessor unusable exception occurs if an attempt is made to execute a coprocessor instruction or access a system register of the coprocessor without having the corresponding coprocessor use permission (PSW.CUn = 0).

2.4.4 System Registers

Some coprocessor functions are defined by system registers. The coprocessor use permission is necessary to access the system register of a coprocessor function. For some system registers, the supervisor privilege (SV permission) is necessary in addition to the coprocessor use permission.

For details about the permissions necessary to access system registers, see Section 2.5, Registers.



2.5 Registers

This CPU defines program registers (general-purpose registers and the program counter PC) and system registers for controlling the status and storing exception information.

2.5.1 **Program Registers**

The program registers include general-purpose registers (r0 to r31) and the program counter (PC).

| Category | Access Permission | Name |
|---------------------------|-------------------|-----------|
| Program counter | UM | PC |
| General-purpose registers | UM | r0 to r31 |

Note: UM: User register. This register can always be accessed because no access permission is required.

2.5.2 System Registers

For details about program registers, see Section 3.1, Program Registers.

Group numbers 0 to 3 : Registers related to basic functions Group numbers 4 to 7 : Registers related to the memory management function Group numbers 12 to 15 : Registers defined in the CPU hardware specifications Group numbers 16 and later : Reserved for future expansion

For details about system registers, see the relevant sections in Section 3, REGISTER SET.

2.5.3 Register Updating

There are several methods used to update registers. Normally, no particular restrictions apply when updating register by using an instruction. However, when updating registers by using the following instructions, some restrictions might apply, depending on the operating mode.

- LDSR
- STSR



(1) LDSR and STSR

The LDSR and STSR instructions can access all the system registers. However, If a system register is accessed without the proper permission, a PIE exception or UCPOP exception might occur. For details about the access permission for each register, see the description of system registers in **Section 3**, **REGISTER SET**. For details about behaviors when a privilege violation occurs, see **Section 2.1.3**, **CPU Operating Modes and Privileges**.

Figure 2.3 shows the flow of executing the LDSR and STSR instructions.



Figure 2.3 Flow of Executing the LDSR and STSR Instructions



2.5.4 Accessing Undefined Registers

If a system register number without any register assigned is accessed or if an inaccessible register is accessed, the following results occur.

- Undefined registers are handled as having the SV permission. When they are accessed by an LDSR or STSR instruction in user mode (PSW.UM = 1), a PIE exception occurs.
- For a read operation, the read result is undefined. If the read value is used in a program, unexpected behaviors might occur.
- For a write operation, the write operation is ignored. However, writing to the following system register numbers is prohibited. Writing prohibited: [SR11, 0], [SR1, 1], [SR7, 1], [SR10, 1], [SR13, 1], [SR14, 1], [SR15, 1], [SR16, 1], [SR5, 2], [SR20, 5]



2.6 Data Types

2.6.1 Data formats

This CPU handles data in little endian format. This means that byte 0 of a halfword or a word is always the least significant (rightmost) byte.

The supported data format is as follows.

- Byte (8-bit data)
- Halfword (16-bit data)
- Word (32-bit data)
- Double-word (64-bit data)
- Bit (1-bit data)

(1) Byte

A byte is 8 consecutive bits of data that starts from any byte boundary. Numbers from 0 to 7 are assigned to these bits, with bit 0 as the LSB (least significant bit) and bit 7 as the MSB (most significant bit). The byte address is specified as "A".



(2) Halfword

A halfword is two consecutive bytes (16 bits) of data that starts from any byte boundary. Numbers from 0 to 15 are assigned to these bits, with bit 0 as the LSB and bit 15 as the MSB. The bytes in a halfword are specified using address "A", so that the two addresses comprise byte data of "A" and "A + 1".





(3) Word

A word is four consecutive bytes (32 bits) of data that starts from any byte boundary. Numbers from 0 to 31 are assigned to these bits, with bit 0 as the LSB (least significant bit) and bit 31 as the MSB (most significant bit). A word is specified by address "A" and consists of byte data of four addresses: "A", "A + 1", "A + 2", and "A + 3".



(4) Double-word

A double-word is eight consecutive bytes (64 bits) that start from any 4-byte boundary. Numbers from 0 to 63 are assigned to these bits, with bit 0 as the LSB and bit 63 as the MSB. A double-word is specified by address "A" and consists of byte data of eight addresses: "A", "A + 1", "A + 2", "A + 3", "A + 4", "A + 5", "A + 6", and "A + 7".



(5) Bit

A bit is bit data at the nth bit within 8-bit data that starts from any byte boundary. Each bit is specified using its byte address "A" and its bit number "n" (n = 0 to 7).





2.6.2 Data Representation

(1) Integers

Integers are represented as binary values using 2's complement, and are used in one of four lengths: 64 bits, 32 bits, 16 bits, or 8 bits. Regardless of the length of an integer, its place uses bit 0 as the LSB, and this place gets higher as the bit number increases. Because this is a 2's complement representation, the MSB is used as a signed bit.

The integer ranges for various data lengths are as follows.

- Double-word (64 bits) : -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
- Word (32 bits) : -2,147,483,648 to +2,147,483,647
- Halfword (16 bits) : -32,768 to +32,767
- Byte (8 bits) : -128 to +127

(2) Unsigned integers

In contrast to "integers" which are data that can take either a positive or negative sign, "unsigned integers" are never negative integers. Like integers, unsigned integers are represented as binary values, and are used in one of four lengths: 64 bits, 32 bits, 16 bits, or 8 bits. Also like integers, the place of unsigned integers uses bit 0 as the LSB and gets higher as the bit number increases. However, unsigned integers do not use a sign bit.

The unsigned integer ranges for various data lengths are as follows.

- Double-word (64 bits): 0 to 18,446,744,073,709,551,615
- Word (32 bits): 0 to 4,294,967,295
- Halfword (16 bits): 0 to 65,535
- Byte (8 bits): 0 to 255

(3) Bits

Bit data are handled as single-bit data with either of two values: cleared (0) or set (1). There are four types of bit-related operations (listed below), which target only single-byte data in the memory space.

- Set
- Clear
- Invert
- Test



2.6.3 Data Alignment

When the result of address calculation is a misaligned address, a misaligned access exception (MAE) occurs.

When the data to be processed is in halfword format, misaligned access indicates the access to an address that is not at the halfword boundary (where the address LSB = 0), and when the data to be processed is in word format, misaligned access indicates the access to an address that is not at the word boundary (where the lower two bits of the address = 0). When the data to be processed is in double-word format, misaligned access indicates the access to an address that is not at the double-word boundary (where the lower 3 bits of the address = 0).

For the double-word format only, a misaligned access exception does not occur when data is placed at the word boundary rather than the double-word boundary.

CAUTIONS

- 1. The following instructions might possibly cause misaligned access. For details, see the relevant descriptions in Section 7, INSTRUCTION.
 - LD.H, LD.HU, LD.W, LD.DW
 - SLD.H, SLD.HU, SLD.W
 - ST.H, ST.W, ST.DW
 - SST.H, SST.W
 - LDL.W, STC.W, CAXI
- 2. The following instructions do not cause misaligned access, because the address is rounded in the instruction specification when the alignment specification is incorrect.
 - PREPARE, DISPOSE
 - PUSHSP, POPSP







2.7 Address Space

This CPU supports a linear address space of up to 4 Gbytes. Both memory and I/O can be mapped to this address space (using the memory mapped I/O method). The CPU outputs a 32-bit address for memory and I/O, in which the highest address number is " $2^{32} - 1$ ".

The byte data placed at various addresses is defined with bit 0 as the LSB and bit 7 as the MSB. When the data is comprised of multiple bytes, it is defined so that the byte data at the lowest address is the LSB and the byte data at the highest address is the MSB (i.e., in little endian format).

This manual stipulates that, when representing data comprised of multiple bytes, the right edge must be represented as the lower address and the left side as the upper address, as shown below.



Figure 2.5 Address Space Byte Format



2.7.1 Memory Map

This CPU is 32-bit architecture and supports a linear address space of up to 4 Gbytes. The whole range of this 4-Gbyte address space can be addressed by instruction addressing (instruction access) and operand addressing (data access).

A memory map is shown in **Figure 2.6**.



Figure 2.6 Memory Map (Address Space)



2.7.2 Instruction Addressing

The instruction address is determined based on the contents of the program counter (PC), and is automatically incremented according to the number of bytes in the executed instruction. When a branch instruction is executed, the addressing shown below is used to set the branch destination address to the PC.

(1) Relative addressing (PC relative)

Signed N-bit data (displacement: disp N) is added to the instruction code in the program counter (PC). In this case, displacement is handled as 2's complement data, and the MSB is a signed bit (S). If the displacement is less than 32 bits, the higher bits are sign-extended (N differs from one instruction to another).

The JARL, JR, and Bcond instructions are used with this type of addressing.



Figure 2.7 Relative Addressing

(2) Register addressing (register indirect)

The contents of the general-purpose register (reg1) or system register (regID) specified by the instruction are transferred to the program counter (PC).

The JMP, CTRET, EIRET, FERET, and DISPOSE instructions are used with this type of addressing.



Figure 2.8 Register Addressing

(3) Based addressing

Contents that are specified by the instruction in the general-purpose register (reg1) and that include the added N-bit displacement (dispN) are transferred to the program counter (PC). At this time, the displacement is handled as a 2's complement data, and the MSB is a signed bit (S). If the displacement is less than 32 bits, the higher bits are sign-extended (N differs from one instruction to another).

The JMP instruction is used with this type of addressing.



Figure 2.9 Based Addressing

(4) Other addressing

A value specified by an instruction is transferred to the program counter (PC). How a value is specified is explained in [Operation] or [Description] of each instruction.

The CALLT, SYSCALL, TRAP, FETRAP, and RIE instructions, and branch in case of an exception are used with this type of addressing.



2.7.3 Data Addressing

The following methods can be used to access the target registers or memory when executing an instruction.

(1) Register addressing

This addressing method accesses the general-purpose register or system register specified in the general-purpose register field as an operand.

Any instruction that includes the operand reg1, reg2, reg3, or regID is used with this type of addressing.

(2) Immediate addressing

This address mode uses arbitrary size data as the operation target in the instruction code.

Any instruction that includes the operand imm5, imm16, vector, or cccc is used with this type of addressing.

NOTE

- vector : This is immediate data that specifies the exception vector (00_H to $1F_H$), and is an operand used by the TRAP, FETRAP, and SYSCALL instructions. The data width differs from one instruction to another.
- cccc : This is 4-bit data that specifies a condition code, and is an operand used in the CMOV instruction, SASF instruction, and SETF instruction. One bit (0) is added to the higher position and is then assigned to an opcode as a 5-bit immediate data.

(3) Based addressing

There are two types of based addressing, as described below.

(a) Type 1

The contents of the general-purpose register (reg1) specified at the addressing specification field in the instruction code are added to the N-bit displacement (dispN) data sign-extended to word length to obtain the operand address, and addressing accesses the target memory for the operation. At this time, the displacement is handled as a 2's complement data, and the MSB is a signed bit (S). If the displacement is less than 32 bits, the higher bits are sign-extended (N differs from one instruction to another).

The LD, ST, and CAXI instructions are used with this type of addressing.





(b) Type 2

This addressing accesses a memory to be manipulated by using as an operand address the sum of the contents of the element pointer (r30) and N-bit displacement data (dispN) that is zeroextended to a word length. If the displacement is less than 32 bits, the higher bits are signextended (N differs from one instruction to another).

The SLD instruction and SST instruction are used with this type of addressing.



Figure 2.11 Based Addressing (Type 2)

(4) Bit addressing

The contents of the general-purpose register (reg1) are added to the N-bit displacement (dispN) data sign-extended to word length to obtain the operand address, and bit addressing accesses one bit (as specified by 3-bit data "bit #3") in one byte of the target memory space. At this time, the displacement is handled as a 2's complement data, and the MSB is a signed bit (S). If the displacement is less than 32 bits, the higher bits are sign-extended (N differs from one instruction to another).

The CLR1, SET1, NOT1, and TST1 instructions are used with this type of addressing.



Figure 2.12 Bit Addressing



(5) Post index increment/decrement addressing

The contents of the general-purpose register (reg1) are used as an operand address to access the target memory, and then the general-purpose register (reg1) is updated. The register is updated by either incrementing or decrementing it, and there are three types (1 to 3).

If the result of incrementing the general-purpose register (reg1) value exceeds the positive maximum value 0xFFFF FFFF, the result wraps around to 0x0000 0000, and, if the result of decrementing the general-purpose register value is less than the positive minimum value 0x0000 0000, the result wraps around to 0xFFFF FFFF.

(a) Type 1

The general-purpose register (reg1) is updated by adding a constant that depends on the type of accessed data (the size of the accessed data) to the contents of the general-purpose register (reg1). If the type of accessed data is a byte, 1 is added, if the type is a halfword, 2 is added, if the type is a word, 4 is added, and if the type is a double-word, 8 is added.



Figure 2.13 Post Index Increment/Decrement Addressing (Type 1)



(b) Type 2

The general-purpose register (reg1) is updated by subtracting a constant that depends on the size of the accessed data from the contents of the general-purpose register (reg1). If the size of accessed data is a byte, 1 is subtracted, if the size is a halfword, 2 is subtracted, if the size is a word, 4 is subtracted, and if the size is a double-word, 8 is subtracted.



Figure 2.14 Post Index Increment/Decrement Addressing (Type 2)

(c) Type 3

The general-purpose register (reg1) is updated by adding the contents of another general-purpose register (reg2) to it. If the MSB of the general-purpose register (reg2) is 1, a negative value is indicated, so a post decrement operation is performed. If this MSB is 0, a positive value is indicated, so a post increment operation is performed. The value of the general-purpose register (reg2) does not change.



Figure 2.15 Post Index Increment/Decrement Addressing (Type 3)



(6) Other addressing

This addressing is to access a memory to be manipulated by using a value specified by an instruction as the operand address. How a value is specified is explained in [Operation] or [Description] of each instruction.

The SWITCH, CALLT, SYSCALL, PREPARE, DISPOSE, PUSHSP, and POPSP instructions are used with this type of addressing.



2.8 Acquiring the CPU Number

This CPU provides a method for identifying CPUs in a multi-processor system.

In the multi-processor configuration, you can identify which CPU core is running a program by referencing HTCFG0.PEID. With HTCFG0.PEID, unique numbers are assigned within multi-processor systems.

2.9 System Protection Identifier

In this CPU, memory resources and peripheral devices are managed by system protection groups. By specifying the group to which the program being executed belongs, you can assign operable memory resources and peripheral devices to each machine.

The program being executed belongs to the group shown by MCFG0.SPID, and whether the memory resources and peripheral devices are operable is decided using this SPID. Any value can be set to MCFG0.SPID by the supervisor.

CAUTION

According to the value of MCFG0.SPID, how operations are assigned to memory resources and peripheral devices is determined by the hardware specifications.



Section 3 REGISTER SET

This chapter describes the program register and system register mounted on this CPU.

3.1 Program Registers

Program registers includes general-purpose registers (r0 to r31) and the program counter (PC). r0 always retains 0, whereas the value after reset is undefined in r1 to r31.

| Program Register | Name | Function | Description | |
|------------------|-----------|------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|--|
| General-purpose | r0 | Zero register | Always retains 0 | |
| registers | r1 | Assembler reserved register | Used as working register for generating addresses | |
| | r2 | Register for address and data variables (used when the real-time OS used does not use this register) | | |
| | r3 | Stack pointer (SP) | Used for generating a stack frame when a function is called | |
| | r4 | Global pointer (GP) | Used for accessing a global variable in the data area | |
| | r5 | Text pointer (TP) | Used as a register that indicates the start of the text area (area where program code is placed) | |
| | r6 to r29 | Register for addresses and data variables | | |
| | r30 | Element pointer (EP) | Used as a base pointer for generating addresses when accessing memory | |
| | r31 | Link pointer (LP) | Used when the compiler calls a function | |
| Program counter | PC | Retains instruction addresses during execution of programs | | |

Table 3.1 Program Registers

Note: For further descriptions of r1, r3 to r5, and r31 used for an assembler and/or C compiler, see the manual of each software development environment.


3.1.1 General-Purpose Registers

A total of 32 general-purpose registers (r0 to r31) are provided. All of these registers can be used for either data variables or address variables.

Of the general-purpose registers, r0 to r5, r30, and r31 are assumed to be used for special purposes in software development environments, so it is necessary to note the following when using them.

(a) r0, r3, and r30

These registers are implicitly used by instructions.

r0 is a register that always retains 0. It is used for operations that use 0, addressing with base address being 0, etc.

r3 is implicitly used by the PREPARE, DISPOSE, PUSHSP, and POPSP instructions.

r30 is used as a base pointer when the SLD instruction or SST instruction accesses memory.

(b) r1, r4, r5, and r31

These registers are implicitly used by the assembler and C compiler.

When using these registers, register contents must first be saved so they are not lost and can be restored after the registers are used.

(c) r2

This register is used by a real-time OS in some cases. If the real-time OS that is being used is not using r2, r2 can be used as a register for address variables or data variables.



3.1.2 PC — Program Counter

The PC retains the address of the instruction being executed.



| Table 3.2 PC Register Content | Table 3.2 | PC Registe | r Contents |
|-------------------------------|-----------|------------|------------|
|-------------------------------|-----------|------------|------------|

| Bit | Name | Description | R/W | Value after Reset |
|---------|-------------|-------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | PC31 to PC1 | These bits indicate the address of the instruction being executed. | R/W | *1 |
| 0 | PC0 | This bit is fixed to 0. Branching to an odd number address is disabled. | R/W | 0 |

Note 1. For details, see the hardware manual of the product used.



3.2 Basic System Registers

The basic system registers are used to control CPU status and to retain exception information.

Basic system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

 Table 3.3
 Basic System Registers (1/2)

| Register No. (regID, seIID) | Symbol | Function | Access Permission |
|--------------------------------|--------|------------------------------------------------------------------------|-------------------|
| SR0, 0 | EIPC | Status save registers when acknowledging EI level exception | SV |
| SR1, 0 | EIPSW | Status save registers when acknowledging EI level exception | SV |
| SR2, 0 | FEPC | Status save registers when acknowledging FE level exception | SV |
| SR3, 0 | FEPSW | Status save registers when acknowledging FE level exception | SV |
| SR5, 0 | PSW | Program status word | *1 |
| SR6, 0 | FPSR | (See Section 3.4, FPU Function Registers) | CU0 and SV |
| SR7, 0 | FPEPC | (See Section 3.4, FPU Function Registers) | CU0 and SV |
| SR8, 0 | FPST | (See Section 3.4, FPU Function Registers) | CU0 |
| SR9, 0 | FPCC | (See Section 3.4, FPU Function Registers) | CU0 |
| SR10, 0 | FPCFG | (See Section 3.4, FPU Function Registers) | CU0 |
| SR13, 0 | EIIC | El level exception cause | SV |
| SR14, 0 | FEIC | FE level exception cause | SV |
| SR16, 0 | CTPC | CALLT execution status save register | UM |
| SR17, 0 | CTPSW | CALLT execution status save register | UM |
| SR20, 0 | СТВР | CALLT base pointer | UM |
| SR28, 0 | EIWR | EI level exception working register | SV |
| SR29, 0 | FEWR | FE level exception working register | SV |
| SR31, 0 | (BSEL) | (Reserved for backward compatibility with V850E2 series)* ² | SV |
| SR0, 1 | MCFG0 | Machine configuration | SV |
| SR2, 1 | RBASE | Reset vector base address | SV |
| SR3, 1 | EBASE | Exception handler vector address | SV |
| SR4, 1 | INTBP | Base address of the interrupt handler table | SV |
| SR5, 1 | MCTL | CPU control | SV |
| SR6, 1 | PID | Processor ID | SV |
| SR11, 1 | SCCFG | SYSCALL operation setting | SV |
| SR12, 1 | SCBP | SYSCALL base pointer | SV |
| SR0, 2 | HTCFG0 | Thread configuration | SV |
| SR6, 2 | MEA | Memory error address | SV |
| SR7, 2 | ASID | Address space ID | SV |
| SR8, 2 | MEI | Memory error information | SV |



- Note 1. The access permission differs depending on the bit. For details, see (5), PSW Program status word in Section 3.2, Basic System Registers.
- Note 2. This bit is reserved to maintain backward compatibility with V850E2 series. This bit is always 0 when read. Writing to this bit is ignored.

(1) EIPC — Status save register when acknowledging EI level exception

When an EI level exception is acknowledged, the address of the instruction that was being executed when the EI level exception occurred, or of the next instruction, is saved to the EIPC register (see **Section 4.1.3, Types of Exceptions**).

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the EIPC register. An odd-numbered address must not be specified.



Table 3.4EIPC Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | EIPC31 to EIPC1 | These bits indicate the PC saved when an EI level exception is acknowledged. | R/W | Undefined |
| 0 | EIPC0 | This bit indicates the PC saved when an EI level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the EIRET instruction is executed is 0. | R/W | Undefined |



(2) EIPSW — Status save register when acknowledging EI level exception

When an EI level exception is acknowledged, the current PSW setting is saved to the EIPSW register.

Because there is only one pair of EI level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

CAUTION

Bits 11 to 9 are related to the debug function and therefore cannot normally be changed.



| Table 2.5 | | Pagistar | Contonto |
|-----------|-------|----------|----------|
| Table 3.5 | EIPOW | Register | Contents |

| Bit | Name | Description | R/W | Value after Reset |
|----------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 30 | UM | This bit stores the PSW.UM bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 29 to 19 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 18 to 16 | CU2 to CU0 | These bits store the PSW.CU2-0 field setting when an EI level exception is acknowledged. (CU2-1 are reserved for future expansion. Be sure to set to 0.) | R/W | 0 |
| 15 | EBV | This bit stores the PSW.EBV bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 14 to 12 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 11 to 9 | Debug | These bits store the PSW.Debug field setting when an EI level exception is acknowledged. | R/W | 0 |
| 8 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | NP | This bit stores the PSW.NP bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 6 | EP | This bit stores the PSW.EP bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 5 | ID | This bit stores the PSW.ID bit setting when an EI level exception is acknowledged. | R/W | 1 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when an EI level exception is acknowledged. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when an EI level exception is acknowledged. | R/W | 0 |



(3) FEPC — Status save register when acknowledging FE level exception

When an FE level exception is acknowledged, the address of the instruction that was being executed when the FE level exception occurred, or of the next instruction, is saved to the FEPC register (see **Section 4.1.3, Types of Exceptions**). Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

Be sure to set an even-numbered address to the FEPC register. An odd-numbered address must not be specified.



| Table 3.6 | FFPC | Register | Contents |
|------------|------|----------|----------|
| 1 abie 5.0 | FEFC | Register | Contents |

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | FEPC31 to FEPC1 | These bits indicate the PC saved when an FE level exception is acknowledged. | R/W | Undefined |
| 0 | FEPC0 | This bit indicates the PC saved when an FE level exception is acknowledged. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the FERET instruction is executed is 0. | R/W | Undefined |



(4) FEPSW — Status save register when acknowledging FE level exception

When an FE level exception is acknowledged, the current PSW setting is saved to the FEPSW register.

Because there is only one pair of FE level exception status save registers, when processing multiple exceptions, the contents of these registers must be saved by a program.

CAUTION

Bits 11 to 9 are related to the debug function and therefore cannot normally be changed.



| | | | _ |
|-----------|-------|----------|----------|
| Table 3.7 | FFPSW | Ronistor | Contente |
| | | NEGISIEI | OOntento |

| Bit | Name | Description | R/W | Value after Reset |
|----------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 30 | UM | This bit stores the PSW.UM bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 29 to 19 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 18 to 16 | CU2 to CU0 | These bits store the PSW.CU2-0 field setting when an FE level exception is acknowledged. (CU2-1 are reserved for future expansion. Be sure to set to 0.) | R/W | 0 |
| 15 | EBV | This bit stores the PSW.EBV bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 14 to 12 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 11 to 9 | Debug | These bits store the PSW.Debug field setting when an FE level exception is acknowledged. | R/W | 0 |
| 8 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | NP | This bit stores the PSW.NP bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 6 | EP | This bit stores the PSW.EP bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 5 | ID | This bit stores the PSW.ID bit setting when an FE level exception is acknowledged. | R/W | 1 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when an FE level exception is acknowledged. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when an FE level exception is acknowledged. | R/W | 0 |



(5) PSW — Program status word

PSW (program status word) is a set of flags that indicate the program status (instruction execution result) and bits that indicate the operation status of the CPU (flags are bits in the PSW that are referenced by a condition instruction (Bcond, CMOV, etc.)).

CAUTIONS

- 1. When the LDSR instruction is used to change the contents of bits 7 to 0 in this register, the changed contents become valid from the instruction following the LDSR instruction.
- 2. The access permission for the PSW register differs depending on the bit. All bits can be read, but some bits can only be written under certain conditions. See Table 3.8 for the access permission for each bit.

| | | 3 | |
|----------|------------|-----------------------------------|--------------------------------|
| Bit | | Access Permission When Reading | Access Permission When Writing |
| 30 | UM | UM | SV*1 |
| 19 | HVC | | Special ^{*1} |
| 18 to 16 | CU2 to CU0 | | SV*1 |
| 15 | EBV | | SV*1 |
| 11 to 9 | Debug | | Special ^{*1} |
| 7 | NP | | SV ^{*1} |
| 6 | EP | | SV*1 |
| 5 | ID | | SV*1 |
| 4 | SAT | | UM |
| 3 | CY | | UM |
| 2 | OV | | UM |
| 1 | S | | UM |
| 0 | Z | | UM |

 Table 3.8
 Access Permission for PSW Register

Note 1. The access permission for the whole PSW register is UM, so the PIE exception does not occur even if the register is written by using an LDSR instruction when PSW.UM is 1. In this case, writing is ignored.





| Table 3.9 | PSW Register Cor | ntents (1/2) |
|-----------|-------------------------|--------------|
| | | |

| Bit | Name | Description | R/W | Value after Reset |
|----------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 30 | UM | This bit indicates that the CPU is in user mode (in UM mode). 0: Supervisor mode 1: User mode | | 0 |
| 29 to 19 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 18 to 16 | CU2 to CU0 | These bits indicate the coprocessor use permissions. When the bit corresponding to the coprocessor is 0, a coprocessor unusable exception occurs if an instruction for the coprocessor is executed or a coprocessor resource (system register) is accessed. CU2 bit 18: (Reserved for future expansion. Be sure to set to 0.) CU1 bit 17: (Reserved for future expansion. Be sure to set to 0.) CU0 bit 16: FPU | R/W | 000 |
| 15 | EBV | This bit indicates the reset vector and exception vector operation. See the description on RBASE and EBASE in this section. | R/W | 0 |
| 14 to 12 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 11 to 9 | Debug | This bit is used for the debug function for the development tool. Always set this bit to 0. | — | 0 |
| 8 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | NP | This bit disables the acknowledgement of FE level exception. When an FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level and FE level exceptions. As for the exceptions which the NP bit disables the acknowledgment, see Table 4.1, Exception Cause List . 0: The acknowledgement of FE level exception is enabled. 1: The acknowledgement of FE level exception is disabled. | R/W | 0 |
| 6 | EP | This bit indicates that an exception other than an interrupt is being serviced. It is set to 1 when the corresponding exception occurs. This bit does not affect acknowledging an exception request even when it is set to 1. 0: An exception other than an interrupt is not being serviced. 1: An exception other than an interrupt is being serviced. | R/W | 0 |
| 5 | ID | This bit disables the acknowledgement of EI level exception. When an EI level or FE level exception is acknowledged, this bit is set to 1 to disable the acknowledgement of EI level exception. As for the exceptions which the ID bit disables the acknowledgment, see Table 4.1, Exception Cause List . This bit is also used to disable EI level exceptions from being acknowledged as a critical section while an ordinary program or interrupt is being serviced. It is set to 1 when the DI instruction is executed, and cleared to 0 when the EI instruction is executed. The change of the ID bit by the EI or ID instruction will be enabled from the next instruction. 0: The acknowledgement of EI level exception is enabled. 1: The acknowledgement of EI level exception is disabled. | R/W | 1 |
| 4 | SAT* ¹ | This bit indicates that the operation result is saturated because the result of a saturated operation instruction operation has overflowed. This is a cumulative flag, so when the operation result of the saturated operation instruction becomes saturated, this bit is set to 1, but it is not later cleared to 0 when the operation result for a subsequent instruction is not saturated. This bit is cleared to 0 by the LDSR instruction. This bit is neither set to 1 nor cleared to 0 when an arithmetic operation instruction is executed. 0: Not saturated 1: Saturated | R/W | 0 |

RENESAS

| Bit | Name | Description | R/W | Value after Reset |
|-----|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 3 | CY | This bit indicates whether a carry or borrow has occurred in the operation result. 0: Carry and borrow have not occurred. 1: Carry or borrow has occurred. | R/W | 0 |
| 2 | OVNote*1 | This bit indicates whether or not an overflow has occurred during an operation.0: Overflow has not occurred.1: Overflow has occurred. | R/W | 0 |
| 1 | SNote*1 | This bit indicates whether or not the result of an operation is negative. 0: Result of operation is positive or 0. 1: Result of operation is negative. | R/W | 0 |
| 0 | Z | This bit indicates whether or not the result of an operation is 0. 0: Result of operation is not 0. 1: Result of operation is 0. | R/W | 0 |

Table 3.9 PSW Register Contents (2/2)

Note 1. The operation result of the saturation processing is determined in accordance with the contents of the OV flag and S flag during a saturated operation. When only the OV flag is set to 1 during a saturated operation, the SAT flag is set to 1.

| | Flag Status | | | |
|---------------------------------------|----------------|----|---|-------------------------|
| Operation Result Status | SAT | ov | S | Saturation Processing |
| Exceeded positive maximum value | 1 | 1 | 0 | 7FFF FFFF _H |
| Exceeded negative maximum value | 1 | 1 | 1 | 8000 0000 _H |
| Positive (maximum value not exceeded) | Value prior to | 0 | 0 | Operation result itself |
| Negative (maximum value not exceeded) | retained. | | 1 | |

(6) EIIC — EI level exception cause

The EIIC register retains the cause of any EI level exception that occurs. The value retained in this register is an exception code corresponding to a specific exception cause (see **Table 4.1, Exception Cause List**).



Table 3.10EIIC Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | EIIC31 to EIIC0 | These bits store the exception cause code when an EI level exception occurs. The EIIC15-0 field stores the exception cause codes shown in Table 4.1 . The EIIC31-16 field stores detailed exception cause codes defined individually for each exception. If there is no particular definition, these bits are set to 0. | R/W | 0 |



(7) FEIC — FE level exception cause

The FEIC register retains the cause of any FE level exception that occurs. The value retained in this register is an exception code corresponding to a specific exception cause (see **Table 4.1, Exception Cause List**).



| Table 3 11 | FEIC | Register | Contents |
|------------|------|----------|----------|
| | | Negister | Contents |

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | FEIC31 to FEIC0 | These bits store the exception cause code when an FE level exception occurs. The FEIC15-0 field stores the exception cause codes shown in Table 4.1 . The FEIC31-16 field stores detailed exception cause codes defined individually for each exception. If there is no particular definition, these bits are set to 0. | R/W | 0 |

(8) CTPC — Status save register when executing CALLT

When a CALLT instruction is executed, the address of the next instruction after the CALLT instruction is saved to CTPC.

Be sure to set an even-numbered address to the CTPC register. An odd-numbered address must not be specified.



Table 3.12 CTPC Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | CTPC31 to CTPC1 | These bits indicate the PC of the instruction after the CALLT instruction. | R/W | Undefined |
| 0 | CTPC0 | This bit indicates the PC of the instruction after the CALLT instruction. Always set this bit to 0. Even if it is set to 1, the value transferred to the PC when the CTRET instruction is executed is 0. | R/W | Undefined |



(9) CTPSW — Status save register when executing CALLT

When a CALLT instruction is executed, some of the PSW (program status word) settings are saved to CTPSW.



| Table 3.13 | CTPSW | Register | Contents |
|------------|-------|----------|----------|
|------------|-------|----------|----------|

| Bit | Name | Description | R/W | Value after Reset |
|---------|------|---------------------------------------------------------------------------------|-----|----------------------|
| 31 to 5 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 | SAT | This bit stores the PSW.SAT bit setting when the CALLT instruction is executed. | R/W | 0 |
| 3 | CY | This bit stores the PSW.CY bit setting when the CALLT instruction is executed. | R/W | 0 |
| 2 | OV | This bit stores the PSW.OV bit setting when the CALLT instruction is executed. | R/W | 0 |
| 1 | S | This bit stores the PSW.S bit setting when the CALLT instruction is executed. | R/W | 0 |
| 0 | Z | This bit stores the PSW.Z bit setting when the CALLT instruction is executed. | R/W | 0 |

(10) CTBP — CALLT base pointer

The CTBP register is used to specify table addresses of the CALLT instruction and generate target addresses.

Be sure to set the CTBP register to a halfword address.



Table 3.14 CTBP Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | CTBP31 to CTBP1 | These bits indicate the base pointer address of the CALLT instruction. These bits indicate the start address of the table used by the CALLT instruction. | R/W | Undefined |
| 0 | CTBP0 | This bit indicates the base pointer address of the CALLT instruction. These bits indicate the start address of the table used by the CALLT instruction. Always set this bit to 0. | R | 0 |



(11) ASID — Address space ID

This is the address space ID. This is used to identify the address space provided by the memory management function.



Table 3.15 ASID Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|----------|------|-------------------------------------------------------|-----|----------------------|
| 31 to 10 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 9 to 0 | ASID | This is the address space ID. | R/W | Undefined |

(12) EIWR — EI level exception working register

The EIWR register is used as a working register when an EI level exception has occurred.



| Table 3.16 | EIWR | Register | Contents |
|------------|------|----------|----------|
|------------|------|----------|----------|

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | EIWR31 to EIWR0 | These bits constitute a working register that can be used for any purpose during the processing of an EI level exception. Use this register for purposes such as storing the values of general-purpose registers. | R/W | Undefined |

(13) FEWR — FE level exception working register

The FEWR register is used as a working register when an FE level exception has occurred.



| Table 3.17 | FEWR | Register | Contents |
|------------|------|----------|------------|
| | | regiotor | 0011101110 |

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | FEWR31 to FEWR0 | These bits constitute a working register that can be used for any purpose during the processing of an FE level exception. Use this register for purposes such as storing the values of general-purpose registers. | R/W | Undefined |

(14) HTCFG0 — Thread configuration register



Table 3.18 HTCFG0 Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|----------|------|-------------------------------------------------------|-----|----------------------|
| 31 to 19 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 18 to 16 | PEID | These bits indicate the processor element number. | R | *2 |
| 15 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 1 |
| 14 to 0 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

Note 1. The value after reset depends on the hardware specifications. For details, see the hardware manual of the product used.

Note 2. When these bits are read, the CPU processor identifier defined in the product specifications is read. These bits cannot be written. For details, see the hardware manual of the product used.

(15) MEA — Memory error address register



Table 3.19MEA Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|------|--------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | MEA | These bits store the violation address when an MAE (misaligned) or MPU occurs. | R/W | Undefined |



(16) MEI — Memory error information register

This register is used to store information about the instruction that caused the exception when a misaligned (MAE) or memory protection (MDP) exception occurs.



| Bit | Name | Description | R/W | Value after Reset |
|----------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 21 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 20 to 16 | REG | These bits indicate the number of the source or destination register accessed by the instruction that caused the exception. For details, see Table 3.21 | R/W | Undefined |
| 15 to 11 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 10, 9 | DS | These bits indicate the type of data handled by the instruction that caused the exception. ^{Note} 0: Byte (8 bits) 1: Halfword (16 bits) 2: Word (32 bits) 3: Double-word (64 bits) For details, see Table 3.21 | R/W | Undefined |
| 8 | U | This bit indicates the sign extension method of the instruction that caused the exception. 0: Signed 1: Unsigned For details, see Table 3.21 | R/W | Undefined |
| 7, 6 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5 to 1 | ITYPE | These bits indicate the instruction that caused the exception. For details, see Table 3.21 | R/W | Undefined |
| 0 | RW | This bit indicates whether the operation of the instruction that caused the exception was read (Load-memory) or write (Store-memory). 0: Read (Load-memory) 1: Write (Store-memory) For details, see Table 3.21 | R/W | Undefined |

Table 3.20 MEI Register Contents

Note: Even if the data is divided and access is made several times due to the specifications of the hardware, the original data type indicated by the instruction is stored.

Table 3.21 Instructions Causing Exceptions and Values of MEI Register (1/2)

| Instruction | REG | DS | U | RW | ITYPE |
|----------------|-----|---------------|--------------|-----------|--------|
| SLD.B | dst | 0 (Byte) | 0 (Signed) | 0 (Read) | 00000b |
| SLD.BU | dst | 0 (Byte) | 1 (Unsigned) | 0 (Read) | 00000b |
| SLD.H | dst | 1 (Half-word) | 0 (Signed) | 0 (Read) | 00000b |
| SLD.HU | dst | 1 (Half-word) | 1 (Unsigned) | 0 (Read) | 00000b |
| SLD.W | dst | 2 (Word) | 0 (Signed) | 0 (Read) | 00000b |
| SST.B | src | 0 (Byte) | 0 (Signed) | 1 (Write) | 00000b |
| SST.H | src | 1 (Half-word) | 0 (Signed) | 1 (Write) | 00000b |
| SST.W | src | 2 (Word) | 0 (Signed) | 1 (Write) | 00000b |
| LD.B (disp16) | dst | 0 (Byte) | 0 (Signed) | 0 (Read) | 00001b |
| LD.BU (disp16) | dst | 0 (Byte) | 1 (Unsigned) | 0 (Read) | 00001b |
| LD.H (disp16) | dst | 1 (Half-word) | 0 (Signed) | 0 (Read) | 00001b |
| LD.HU (disp16) | dst | 1 (Half-word) | 1 (Unsigned) | 0 (Read) | 00001b |



RH850G3MH Software

| Та | ble 3.21 Instruc | tions Causing Exc | eptions and Va | lues of MEI Regi | ster (2/2) |
|----------------------------|-------------------|-------------------|----------------|------------------------|------------|
| Instruction | REG | DS | U | RW | ITYPE |
| LD.W (disp16) | dst | 2 (Word) | 0 (Signed) | 0 (Read) | 00001b |
| ST.B (disp16) | src | 0 (Byte) | 0 (Signed) | 1 (Write) | 00001b |
| ST.H (disp16) | src | 1 (Half-word) | 0 (Signed) | 1 (Write) | 00001b |
| ST.W (disp16) | src | 2 (Word) | 0 (Signed) | 1 (Write) | 00001b |
| LD.B (disp23) | dst | 0 (Byte) | 0 (Signed) | 0 (Read) | 00010b |
| LD.BU (disp23) | dst | 0 (Byte) | 1 (Unsigned) | 0 (Read) | 00010b |
| LD.H (disp23) | dst | 1 (Half-word) | 0 (Signed) | 0 (Read) | 00010b |
| LD.HU (disp23) | dst | 1 (Half-word) | 1 (Unsigned) | 0 (Read) | 00010b |
| LD.W (disp23) | dst | 2 (Word) | 0 (Signed) | 0 (Read) | 00010b |
| ST.B (disp23) | src | 0 (Byte) | 0 (Signed) | 1 (Write) | 00010b |
| ST.H (disp23) | src | 1 (Half-word) | 0 (Signed) | 1 (Write) | 00010b |
| ST.W (disp23) | src | 2 (Word) | 0 (Signed) | 1 (Write) | 00010b |
| LD.DW (disp23) | dst | 3 (Double-word) | 0 (Signed) | 0 (Read) | 00010b |
| ST.DW (disp23) | src | 3 (Double-word) | 0 (Signed) | 1 (Write) | 00010b |
| LDL.W | dst | 2 (Word) | 0 (Signed) | 0 (Read) | 00111b |
| STC.W | src | 2 (Word) | 0 (Signed) | 1 (Write) | 00111b |
| CAXI | dst | 2 (Word) | 0 (Signed) | 0 (Read) ^{*1} | 01000b |
| SET1 | _ | 0 (Byte) | 0 (Signed) | 0 (Read) ^{*1} | 01001b |
| CLR1 | _ | 0 (Byte) | 0 (Signed) | 0 (Read) ^{*1} | 01001b |
| NOT1 | — | 0 (Byte) | 0 (Signed) | 0 (Read) ^{*1} | 01001b |
| TST1 | _ | 0 (Byte) | 0 (Signed) | 0 (Read) | 01001b |
| PREPARE | src | 2 (Word) | 0 (Signed) | 1 (Write) | 01100b |
| DISPOSE | dst | 2 (Word) | 0 (Signed) | 0 (Read) | 01100b |
| PUSHSP | src | 2 (Word) | 0 (Signed) | 1 (Write) | 01101b |
| POPSP | dst ^{*4} | 2 (Word) | 0 (Signed) | 0 (Read) | 01101b |
| SWITCH | _ | 1 (Half-word) | 0 (Signed) | 0 (Read) | 10000b |
| CALLT | _ | 1 (Half-word) | 1 (Unsigned) | 0 (Read) | 10001b |
| SYSCALL | _ | 2 (Word) | 0 (Signed) | 0 (Read) | 10010b |
| CACHE | _ | _ | _ | 0/1*2 | 10100b |
| Interrupt (table reference | e method) *3 — | 2 (Word) | 0 (Signed) | 0 (Read) | 10101b |

Note 1. This exception occurs when the instruction executes a read access.

Note 2. The value differs depending on the operation.

Note 3. When the interrupt vector of the table reference method is read.

Note 4. When the destination is r3, 0 is stored.

Note: dst: Destination register number, src: Source register number



(17) RBASE — Reset vector base address

This register indicates the reset vector address when there is a reset. If the PSW.EBV bit is 0, this vector address is also used as the exception vector address.



Table 3.22 RBASE Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 9 | RBASE31 to RBASE9 | These bits indicate the reset vector when there is a reset. When PSW.EBV = 0, this address is also used as the exception vector. The RBASE8-0 bits are not assigned as names because these bits are always 0. | R | Note |
| 8 to 1 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | RINT | When the RINT bit is set, the exception handler address for interrupt processing is reduced. See 4.4.1 (1), Direct vector method . This bit is valid when PSW.EBV = 0. | R | Note |

Note 1. The value after reset depends on the hardware specifications. For details, see the hardware manual of the product used.

(18) EBASE — Exception handler vector address

This register indicates the exception handler vector address. This register is valid when the PSW.EBV bit is 1.



Table 3.23 EBASE Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 9 | EBASE31 to EBASE9 | The exception handler routine address is changed to the address resulting from adding the offset address of each exception to the base address specified for this register. The EBASE8-0 bits are not assigned as names because these bits are always 0. | R/W | Undefined |
| 8 to 1 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | RINT | When the RINT bit is set, the exception handler address for interrupt processing is reduced. See 4.4.1 (1), Direct vector method . | R/W | Undefined |



(19) INTBP — Base address of the interrupt handler address table

This register indicates the base address of the table when the table reference method is selected as the interrupt handler address selection method.



Table 3.24 INTBP Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 9 | INTBP31 to INTBP9 | These bits indicate the base pointer address for an interrupt when the table reference method is used. The value indicated by these bits is the first address in the table used to determine the exception handler when the interrupt specified by the table reference method (EIINT0 to EIINT511) is acknowledged. The INTBP8-0 bits are not assigned as names because these bits are always 0. | R/W | Undefined |
| 8 to 0 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |



(20) PID — Processor ID

The PID register retains a processor identifier that is unique to the CPU. The PID register is a read-only register.

CAUTION

The PID register indicates information used to identify the incorporated CPU core and CPU core configuration. Usage such that the software behavior varies dynamically according to the PID register information is not assumed.

Table 3.25PID Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|----------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 24 | PID | Architecture Identifier This identifier indicates the architecture of the processor. | R | *1 |
| 23 to 8 | _ | Function Identifier This identifier indicates the functions of the processor. These bits indicate whether or not functions defined per bit are implemented (1: implemented, 0: not implemented). Bits 23 and 11 : Reserved Bit 10 : Double-precision floating-point operation function Bit 9 : Single-precision floating-point operation function* ¹ Bit 8 : Memory protection unit (MPU) function NOTE If a double-precision floating-point operation function is implemented (when bit 10 is 1), a single-precision floating-point operation function is also always implemented (bit 9 is 1). | R | *1 |
| 7 to 0 | _ | Version Identifier This identifier indicates the version of the processor. | R | *1 |

Note 1. For details, see the hardware manual of the product used.



(21) SCCFG — SYSCALL operation setting

This register is used to set operations related to the SYSCALL instruction. Be sure to set an appropriate value to this register before using the SYSCALL instruction.





| Bit | Name | Description | R/W | Value after Reset |
|---------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 8 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 to 0 | SIZE | These bits specify the maximum number of entries of a table that the SYSCALL instruction references. The maximum number of entries the SYSCALL instruction references is 1 if SIZE is 0, and 256 if SIZE is 255. By setting the maximum number of entries appropriately in accordance with the number of functions branched by the SYSCALL instruction, the memory area can be effectively used. If a vector exceeding the maximum number of entries is specified for the SYSCALL instruction, the first entry is selected. Place an error processing routine at the first entry. | R/W | Undefined |

(22) SCBP — SYSCALL base pointer

The SCBP register is used to specify a table address of the SYSCALL instruction and generate a target address. Be sure to set an appropriate value to this register before using the SYSCALL instruction.

Be sure to set a word address to the SCBP register.



 Table 3.27
 SCBP Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 2 | SCBP31 to SCBP2 | These bits indicate the base pointer address of the SYSCALL instruction. These bits indicate the start address of the table used by the SYSCALL instruction. | R/W | Undefined |
| 1, 0 | SCBP1, SCBP0 | These bits indicate the base pointer address of the SYSCALL instruction. These bits indicate the start address of the table used by the SYSCALL instruction. Always set these bits to 0. | R | 0 |



(23) MCFG0 — Machine configuration

This register indicates the CPU configuration.



Table 3.28 MCFG0 Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|----------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 24 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 23 to 16 | SPID | These bits indicate the system protection number. The SPID bit width depends on the product and the value that can be written might therefore be restricted. For details, see the hardware manual of the product used. | R/W | *1 |
| 15 to 3 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 2 | _ | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 1, 0 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

Note 1. For details, see the hardware manual of the product used.

(24) MCTL — Control of the CPU

This register is used to control the CPU.



| Bit | Name | Description | R/W | Value after Reset |
|---------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 | _ | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 30 to 1 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | UIC | This bit is used to control the interrupt enable/disable operation in user mode. When this bit is set to 1, executing the EI/DI instruction in user mode become possible. | R/W | 0 |



3.3 Interrupt Function Registers

3.3.1 Interrupt Function System Registers

Interrupt function system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

| | _ | | |
|--------------------------------|--------|--------------------------------------|----------------------|
| Register No. (regID, selID) | Symbol | Function | Access Permission |
| SR10, 2 | ISPR | Priority of interrupt being serviced | SV |
| SR11, 2 | PMR | Interrupt priority masking | SV |
| SR12, 2 | ICSR | Interrupt control status | SV |
| SR13, 2 | INTCFG | Interrupt function setting | SV |

Table 3.30 Interrupt Function System Registers



(1) ISPR — Priority of interrupt being serviced

This register holds the priority of the EIINTn interrupt being serviced. This priority value is then used to perform priority ceiling processing when multiple interrupts are generated.



| Bit | Name | Description | | R/W | Value after Reset |
|----------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|------------------------------------|
| 31 to 16 | _ | (Reserved for future expansion. Be sure to set to 0.) | | | 0 |
| 15 to 0 | ISP15 to ISP0 | These bits indicate the a priority ^{*1} that correspon 0 : An interrupt reques relevant bit position 1 : An interrupt reques relevant position is | acknowledgment status of an EIINT <i>n</i> interrupt with a ds to the relevant bit position. t for an interrupt whose priority corresponds to the has not been acknowledged. t for an interrupt whose priority corresponds to the being serviced by the CPU core. | R ^{*3} | 0 |
| | | The bit positions corresp | bond to the following priority levels: | | |
| | | Bit | Priority | | |
| | | 0 | Priority 0 (highest) | | |
| | | 1 | Priority 1 | | |
| | | | | | |
| | | 14 | Priority 14 | | |
| | | 15 | Priority 15 | | |
| | | to the acknowledged into 0 when the EIRET instru- among the ISP15-0 bits 0* ² . While a bit in this register masked. Priority level ju is determining whether t exceptions will not be ac For details, see Section Masking . When performing softwar sure to clear this register | errupt request is automatically set to 1. If PSW.EP is action is executed, the bit with the highest priority that are set (0 is the highest priority) is cleared to er is set to 1, lower priority interrupts (EIINT<i>n</i>) are dgment is therefore not performed when the system o acknowledge an exception, meaning that exhowledged. 4.1.5, Interrupt Exception Priority and Priority are-based priority control using the PMR register, be r by using the INTCFG.ISPC bit. | | |
| | Note 1. | For details, see Section 4 | .1.5, Interrupt Exception Priority and Priority Mas | king. | |
| | Note 2. | Interrupt acknowledgment by setting (1) the INTCFG cases, the INTCFG.ISPC | and auto-updating of values when the EIRET instruction ISPC bit. It is recommended to enable auto-updating bit should be cleared to 0. | on is exe of value | cuted are disab s, so in normal |
| | Note 3. | This is R or R/W, dependin as a read-only (R) register | ng on the setting of the INTCFG.ISPC bit. It is recommon | nended to | o use this regis |

Table 3.31 ISPR Register Contents



(2) PMR — Interrupt priority masking

This register is used to mask the specified interrupt priority.



Table 3.32 PMR Register Contents

| Bit | Name | Description | | | | R/W | Value after Reset |
|----------|-------------|------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|----------------------------------------|-----|----------------------|
| 31 to 16 | _ | (Reserved for | future expansion | n. Be sure to set to 0.) | | R | 0 |
| 15 to 0 | PM15 to PM0 | These bits ma the relevant b 0 : Servicing relevant b 1 : Servicing relevant b The bit positio | hese bits mask an interrupt request with a priority level that corresponds to be relevant bit position. 0 : Servicing of an interrupt request with a priority that corresponds to the relevant bit position is enabled. 1 : Servicing of an interrupt request with a priority that corresponds to the relevant bit position is disabled. The bit positions correspond to the following priority levels: | | | R/W | 0 |
| | | | Bit | Priority | | | |
| | | | 0 | Priority 0 (highest) | | | |
| | | | 1 | Priority 1 | | | |
| | | | | | | | |
| | | | 14 | Priority 14 | | | |
| | | | 15 | Priority 15 and priority 16 (lowest) | | | |
| | | While a bit in t corresponding performed wh | this register is se to that bit are m en the system is | et to 1, interrupts (EIINT <i>n</i>) with th nasked. Priority level judgment is determining whether to acknowl | e priority therefore not edge an | | |

exception, meaning that exceptions will not be acknowledged*¹.

Note 1. Specify the masks by setting the bits to 1 in order from the lowest-priority bit. For example, $FF00_H$ can be set, but $F0F0_H$ or $00FF_H$ cannot.

(3) ICSR — Interrupt control status

This register indicates the interrupt control status in the CPU.



| Table 3.33 | ICSR | Register | Contents |
|------------|------|----------|----------|
|------------|------|----------|----------|

| Bit | Name | Description | R/W | Value after Reset |
|---------|------|-------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | PMEI | This bit indicates that an interrupt (EIINT <i>n</i>) with the priority level masked by the PMR register exists. | R | 0 |

(4) INTCFG — Interrupt function setting

This register is used to specify settings related to the CPU's internal interrupt function.



Table 3.34 INTCFG Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | ISPC | This bit changes how the ISPR register is written. 0: The ISPR register is automatically updated. Updates triggered by the program (via execution of LDSR instruction) are ignored. 1: The ISPR register is not automatically updated. Updates triggered by the program (via execution of LDSR instruction) are performed. If this bit is cleared to 0, the bits of the ISPR register are automatically set to 1 when an interrupt (EIINT<i>n</i>) is acknowledged, and cleared to 0 when the EIRET instruction executed by the program. If this bit is set to 1, the bits of the ISPR register are not updated by an LDSR instruction. In this case, the bits can be updated by an LDSR instruction. In this case, the bits can be updated by an LDSR instruction is executed. In this case updated by an LDSR instruction. In this case, the bits can be updated by an LDSR instruction is executed by the program. If normal cases, the ISPC bit should be cleared. When performing softwarebased priority control, however, set this bit (1) and perform priority control by using the PMR register. | R/W | 0 |



3.4 FPU Function Registers

3.4.1 Floating-Point Registers

The FPU uses the CPU general-purpose registers (r0 to r31). There are no register files used only for floating-point operations.

- Single-precision floating-point instruction: Thirty-two 32-bit registers can be specified. These general-purpose registers correspond to r0 to r31.
- Double-precision floating-point instruction:

Sixteen 64-bit registers can be specified. Paired general-purpose registers are used as register pairs $(\{r1, r0\}, \{r3, r2\} \dots \{r31, r30\})$. Each register pair is specified in the instruction format with an even numbered register. Because r0 is a zero register (always holds 0), in principle $\{r1, r0\}$ cannot be used by a double-precision floating-point instruction.

3.4.2 Floating-Point Function System Registers

The FPU can use the following system registers to control floating-point operations. Floating-point function system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

- FPSR : This register is used to control and monitor exceptions. It also holds the result of compare operations, and sets the FPU operation mode. Its bits are used to set condition code, subnormal number flush enable, rounding mode control, cause, exception enable, and preservation.
- FPEPC : This register stores the program counter value for the instruction where a floating-point operation exception has occurred.
- FPST : This register reflects the contents of the FPSR register bits related to the operation status.
- FPCC : This register reflects the contents of the FPSR.CC(7:0) bits.
- FPCFG : This register reflects the contents of the FPSR register bits related to the operation settings.

| Register No. | | | |
|----------------|--------|----------------------------------------------------|-------------------|
| (regID, seIID) | Symbol | Function | Access Permission |
| SR6, 0 | FPSR | Floating-point operation configuration/status | CU0 and SV |
| SR7, 0 | FPEPC | Floating-point operation exception program counter | CU0 and SV |
| SR8, 0 | FPST | Floating point operation status | CU0 |
| SR9, 0 | FPCC | Floating-point operation comparison result | CU0 |
| SR10, 0 | FPCFG | Floating-point operation configuration | CU0 |

Table 3.35 FPU System Registers



(1) FPSR — Floating-point configuration/status

This register indicates the execution status of floating-point operations and any exceptions that occur.

For details about exception, see Section 6.1.5, Floating-Point Operation Exceptions.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | _ |
|------|-----|-----|--------|----------|-----|-----|-----|------|---------|------|----|----|---------|---------|----------|----|-------------------|
| FPSR | CC7 | CC6 | CC5 | CC4 | CC3 | CC2 | CC1 | CC0 | FN | IF | 1 | 0 | R | М | FS | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | | 0 | ause t | bits (XC | 5) | | | Enab | le bits | (XE) | | F | Preserv | ation b | oits (XP |) | Value after reset |
| | Е | V | Z | 0 | U | Ι | V | Z | 0 | U | I | V | Z | 0 | U | Ι | *1 |

Table 3.36 FPSR Register Contents (1/2)

| Bit | Name | Descri | ption | R/W | Value after Reset | | |
|----------|---------|------------------------------------------------|---------------------------------------------------|------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----|
| 31 to 24 | CC(7:0) | These comparexcept 0: Con 1: Con | are the rison ir the co mparis mparis | R/W | Undefined | | |
| 23 | FN | This bit roundir numbe Flush t | t enabl ng mod r is flus t o Nea | es flush-to-nea le is RN and th shed to the nea rest . | arest mode. When the FN bit is set to 1, if the e operation result is a subnormal number, the arest number. For details, see Section 6.1.9 , | R/W | 0 |
| 22 | IF | This bit operan Flushi i | t accun ds. Foi n g Sul | nulates and inc r details about f onormal Numb | dicates information about the flushing of input flushing subnormal numbers, see Section 6.1.8 , bers . | R/W | 0 |
| 21 | _ | (Reser | ved for | future expansi | ion. Be sure to set to 1.) | R | 0 |
| 20 | _ | (Reser | ved for | future expansi | ion. Be sure to set to 0.) | R | 0 |
| 19, 18 | RM | These mode t | are the hat the | e rounding mod FPU uses for | le control bits. The RM bits define the rounding all floating-point instructions. | R/W | 00 |
| | | 10 | A Bits | | Description | | |
| | | 0 | 0 | RN | Rounds the result to the nearest representable value. If the value is exactly in-between the two nearest representable values, the result is rounded toward the value whose least significant bit is 0. | | |
| | | 0 | 1 | RZ | Rounds the result toward 0. The result is the nearest to the value that does not exceed the absolute value of the result with infinite accuracy. | | |
| | | 1 | 0 | RP | Rounds the result toward +∞. The result is nearest to a value greater than the accurate result with infinite accuracy. | | |
| | | 1 | 1 | RM | Rounds the result toward $-\infty$. The result is nearest to a value less than the accurate result | | |



| Bit | Name | Description | R/W | Value after Reset | | | | |
|----------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------|-------------------------------------------------|-----------------------------|--------------------------------|-----|-----------|
| 17 | FS | This bit enables values that be flushed. If the FS bit is s subnormal numbers are flu operation exception (E). An flushed to 0 with the same numbers either become 0 c | R/W | 1 | | | | |
| | | Operation result that | Round | ing mode a | and value aft | er flushing | | |
| | | number | RN* ¹ | RZ | RP | RM | | |
| | | Positive | +0 | +0 | +2 ^{Emin} | +0 | | |
| | | Negative | -0 | -0 | -0 | -2 ^{Emin} | | |
| | | Note 1. If the rounding r will occur in the Section 6.1.9, F | node is RI direction of Flush to N | N and the F of higher ac learest . | PSR.FN bit is curacy. For d | s set, flushing etails, see | | |
| 16 | _ | (Reserved for future expansion | sion. Be s | ure to set to | o 0.) | | R | 0 |
| 15 to 10 | XC (E, V, Z, O, U, I) | These are the cause bits. F | or details, | , see 3.4.2 | (1) (a), Cause | e bits (XC). | R/W | Undefined |
| 9 to 5 | XE (V, Z, O, U, I) | These are the enable bits. | R/W | 0 | | | | |
| 4 to 0 | XP (V, Z, O, U, I) | These are the preservation bits (XP) . | R/W | Undefined | | | | |

Table 3.36 FPSR Register Contents (2/2)

Note 1. See the descriptions of each bit.



(a) Cause bits (XC)

Bits 15 to 10 in the FPSR register are cause bits, which indicate the occurrence and cause of a floatingpoint operation exception. If an exception defined by IEEE754 is generated, when an enable bit is set to 1 corresponding to the exception, a cause bit is set, and the exception then occurs. When two or more exceptions occur during a single instruction, each corresponding bit is set to 1.

If two or more exceptions are detected, as long as the enable bit corresponding to one of the exceptions is set to 1, the exception occurs. In this case, the cause bits of all the detected exceptions, including exceptions whose enable bits are cleared to 0, are set to 1.

The cause bits are rewritten by a floating-point instruction (except the TRFSR instruction) where the floating-point operation exception occurred. The E bit is set to 1 when software emulation is required, otherwise it is cleared to 0. Other bits are set to 1 or cleared to 0 depending on whether or not an IEEE754-defined exception has occurred.

When a floating-point operation exception has occurred, the operation result is not stored, and only the cause bits are affected.

When the cause bits are set to 1 by an LDSR instruction, a floating-point operation exception does not occur.

(b) Enable bits (XE)

Bits 9 to 5 in the FPSR register are the enable bits, which enable floating-point operation exceptions. When an IEEE754-defined exception occurs, a floating-point operation exception occurs if the enable bit corresponding to the exception has been set to 1.

There are no enable bits corresponding to an unimplemented operation exception (E). An unimplemented operation exception (E) always occurs as a floating-point operation exception.

If the corresponding enable bit has not been set to 1, no exception occurs and the default result defined by IEEE754 is stored.

(c) Preservation bits (XP)

Bits 4 to 0 in the FPSR register are preservation bits. These bits store and indicate the detected exception after reset. An exception defined by IEEE754 occurs, and if a floating-point operation exception is not generated, the preservation bit is set to 1, otherwise it does not change. The preservation bits are not cleared to 0 by the floating-point operation. However, these bits can be set and cleared by software when an LDSR instruction is used to write a new value to the FPSR register.

There are no preservation bits corresponding to unimplemented operation exceptions (E). An unimplemented operation exception (E) always occurs as a floating-point operation exception.

NOTE

For details about the exception types and how they relate to particular bits, see **Figure 6.6**, **Cause, Enable, and Preservation Bits of FPSR Register**.



(2) FPEPC — Floating-point exception program counter

When an exception that is enabled by an enable bit occurs, the program counter (PC) of the instruction that caused the exception is stored.



Table 3.37 FPEPC Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 1 | FPEPC31 to FPEPC1 | These bits store the program counter (PC) of the floating-point instruction that caused the exception when a floating-point operation exception that is enabled by an enable bit occurs. | R/W | Undefined |
| 0 | FPEPC0 | This bit stores the program counter (PC) of the floating-point instruction that caused the exception when a floating-point operation exception that is enabled by an enable bit occurs. Always set this bit to 0. | R | 0 |

(3) FPST — Floating-point operation status

This register reflects the contents of the FPSR register bits related to the operation status.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | _ |
|------|----|----|----|----|--------------|---------------|---------|----|----|----|----|--------|--------------|---------------|---------------|-------|--------------------------------|
| FPST | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | 0 | 0 | E | V | Cause b Z | oits (XC O | C) U | I | 0 | 0 | IF | V V | Preserv Z | vation t O | oits (XP U |) | Value after reset Undefined |

| Bit | Name | Description | R/W | Reset |
|----------|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----------|
| 31 to 14 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 13 to 8 | XC (E, V, Z, O, U, I) | These are cause bits. For details, see 3.4.2 (1) (a), Cause bits (XC) . Values written to these bits are reflected in FPSR.XC bits. | R/W | Undefined |
| 7, 6 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5 | IF | This bit accumulates and indicates information about the flushing of input operands. For details about flushing subnormal numbers, see Section 6.1.8 , Flushing Subnormal Numbers . | R/W | 0 |
| 4 to 0 | XP (V, Z, O, U, I) | These are preservation bits. For details, see 3.4.2 (1) (c) , Preservation bits (XP) . Values written to these bits are reflected in FPSR.XP bits. | R/W | Undefined |

Table 3.38 FPST Register Contents



Value after

(4) FPCC — Floating-point operation comparison result

This register reflects the contents of the FPSR.CC(7:0) bits.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|------|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|-----|--------------------------------|
| FPCC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CC7 | CC6 | CC5 | CC4 | CC3 | CC2 | CC1 | CC0 | Value after reset Undefined |

| Bit | Name | Description | R/W | Value after Reset |
|---------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 8 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 to 0 | CC (7:0) | These are CC (condition) bits. They store the result of a floating-point comparison instruction. The CC(7:0) bits are not affected by any instructions except the comparison instruction and LDSR instruction. Values written to these bits are reflected in the CC(7:0) bits of FPSR. 0: Comparison result is false 1: Comparison result is true | R/W | Undefined |

Table 3.39 FPCC Register Contents



(5) FPCFG — Floating-point operation configuration

This register reflects the contents of the FPSR register bits related to the operation settings.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|-----------|---------------|-----------|----|---------------------------------------------|
| FPCFG | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | R | M | 0 | 0 | 0 | V | Enat Z | ole bits O | (XE) U | | Value after reset 0000 0000 _H |

| S | |
|---|---|
| 1 | S |

| Bit | Name | Descr | iption | R/W | Value after Reset | | |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------|---------------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|-----|---|
| 31 to 10 | | (Rese | rved for | future expansi | ion. Be sure to set to 0.) | R | 0 |
| 9, 8 | RM | These mode these | are rou that the bits are | R/W | 0 | | |
| | | R | M Bits | | | | |
| | | 9 | 8 | Mnemonic | Description | | |
| | 0 0 RN Rounds the result to the nearest representable value. If the value is exactly in-between the two representable values, the result is rounded toward the value whose least significant bit is 0. | | | | | | |
| | | 0 | 1 | RZ | Rounds the result toward 0. The result is the nearest to the value that does not exceed the absolute value of the result with infinite accuracy. | | |
| | | 1 | 0 | RP | Rounds the result toward +∞. The result is nearest to a value greater than the accurate result with infinite accuracy. | | |
| | | 1 | 1 | RM | Rounds the result toward -∞. The result is nearest to a value less than the accurate result with infinite accuracy. | | |
| | | | | | | | |
| 7 to 5 | — | (Rese | rved for | future expansi | ion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | 4 to 0 XE These are the enable bits. For details, see 3.4.2 (1) (b), Enable bits (XE). (V, Z, O, U, I) Values written to these bits are reflected in the FPSR.XE bits. | | | | | R/W | 0 |



3.5 MPU Function Registers

3.5.1 MPU Function System Registers

MPU function system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

 Table 3.41
 MPU Function System Registers (1/2)

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|--------------------------------|--------|------------------------------------------|-------------------|
| SR0, 5 | MPM | Memory protection operation mode setting | SV |
| SR1, 5 | MPRC | MPU region control | SV |
| SR4, 5 | MPBRGN | MPU base region number | SV |
| SR5, 5 | MPTRGN | MPU end region number | SV |
| SR8, 5 | MCA | Memory protection setting check address | SV |
| SR9, 5 | MCS | Memory protection setting check size | SV |
| SR10, 5 | MCC | Memory protection setting check command | SV |
| SR11, 5 | MCR | Memory protection setting check result | SV |
| SR0, 6 | MPLA0 | Protection area minimum address | SV |
| SR1, 6 | MPUA0 | Protection area maximum address | SV |
| SR2, 6 | MPAT0 | Protection area attribute | SV |
| SR4, 6 | MPLA1 | Protection area minimum address | SV |
| SR5, 6 | MPUA1 | Protection area maximum address | SV |
| SR6, 6 | MPAT1 | Protection area attribute | SV |
| SR8, 6 | MPLA2 | Lower address of the protection area | SV |
| SR9, 6 | MPUA2 | Protection area maximum address | SV |
| SR10, 6 | MPAT2 | Protection area attribute | SV |
| SR12, 6 | MPLA3 | Protection area minimum address | SV |
| SR13, 6 | MPUA3 | Protection area maximum address | SV |
| SR14, 6 | MPAT3 | Protection area attribute | SV |
| SR16, 6 | MPLA4 | Protection area minimum address | SV |
| SR17, 6 | MPUA4 | Protection area maximum address | SV |
| SR18, 6 | MPAT4 | Protection area attribute | SV |
| SR20, 6 | MPLA5 | Protection area minimum address | SV |
| SR21, 6 | MPUA5 | Protection area maximum address | SV |
| SR22, 6 | MPAT5 | Protection area attribute | SV |
| SR24, 6 | MPLA6 | Protection area minimum address | SV |
| SR25, 6 | MPUA6 | Protection area maximum address | SV |
| SR26, 6 | MPAT6 | Protection area attribute | SV |
| SR28, 6 | MLUA7 | Protection area minimum address | SV |
| SR29, 6 | MPUA7 | Protection area maximum address | SV |
| SR30, 6 | MPAT7 | Protection area attribute | SV |
| SR0, 7 | MPLA8 | Protection area minimum address | SV |
| SR1, 7 | MPUA8 | Protection area maximum address | SV |
| SR2, 7 | MPAT8 | Protection area attribute | SV |
| SR4, 7 | MPLA9 | Protection area minimum address | SV |
| SR5, 7 | MPUA9 | Protection area maximum address | SV |



| Register No. (regID, selID) | Symbol | Function | Access Permission |
|--------------------------------|--------|---------------------------------|-------------------|
| SR6, 7 | MPAT9 | Protection area attribute | SV |
| SR8, 7 | MPLA10 | Protection area minimum address | SV |
| SR9, 7 | MPUA10 | Protection area maximum address | SV |
| SR10, 7 | MPAT10 | Protection area attribute | SV |
| SR12, 7 | MPLA11 | Protection area minimum address | SV |
| SR13, 7 | MPUA11 | Protection area maximum address | SV |
| SR14, 7 | MPAT11 | Protection area attribute | SV |
| SR16, 7 | MPLA12 | Protection area minimum address | SV |
| SR17, 7 | MPUA12 | Protection area maximum address | SV |
| SR18, 7 | MPAT12 | Protection area attribute | SV |
| SR20, 7 | MPLA13 | Protection area minimum address | SV |
| SR21, 7 | MPUA13 | Protection area maximum address | SV |
| SR22, 7 | MPAT13 | Protection area attribute | SV |
| SR24, 7 | MPLA14 | Protection area minimum address | SV |
| SR25, 7 | MPUA14 | Protection area maximum address | SV |
| SR26, 7 | MPAT14 | Protection area attribute | SV |
| SR28, 7 | MPLA15 | Protection area minimum address | SV |
| SR29, 7 | MPUA15 | Protection area maximum address | SV |
| SR30, 7 | MPAT15 | Protection area attribute | SV |

Table 3.41 MPU Function System Registers (2/2)

Note: The number of incorporated MPLAn, MPUAn, and MPATn (n = 0 to 15) registers depends on the hardware specifications. For details, see the hardware manual of the product used.



(1) MPM — Memory protection operation mode

The memory protection mode register is used to define the basic operating state of the memory protection function.



| Bit | Name | Description | R/W | Value after Reset | | |
|----------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------|-----------------------------------|--|--|
| 31 to 11 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 | | |
| 10 | DX | This bit specifies the default operation when an instruction is executed at an address that does not exist in a protection area. "0" is fixed for this bit in this CPU. Default operation is prohibited. Be sure to set to 0. 0 : Disable executing an instruction at an address that does not exist in a protection area. 1 : Enable executing an instruction at an address that does not exist in a protection area. The setting of this bit affects the access operation when the protection areas overlap. For details, see Section 5.1.3, Caution Points for Protection Area Setup. | R | 0 | | |
| 9 | DW | This bit specifies the default operation when writing to an address that does not exist in a protection area. "0" is fixed for this bit in this CPU. Default operation is prohibited. Be sure to set to 0. 0: Disable writing to an address that does not exist in a protection area. 1: Enable writing to an address that does not exist in a protection area. The setting of this bit affects the access operation when the protection areas overlap. For details, see Section 5.1.3, Caution Points for Protection Area Setup. | R | 0 | | |
| 8 | DR | This bit specifies the default operation when reading from an address that does not exist in a protection area. "0" is fixed for this bit in this CPU. Default operation is prohibited. Be sure to set to 0. 0 : Disable reading from an address that does not exist in a protection area. 1 : Enable reading from an address that does not exist in a protection area. The setting of this bit affects the access operation when the protection areas overlap. For details, see Section 5.1.3, Caution Points for Protection Area Setup. | R | 0 | | |
| 7 to 2 | — | (Reserved for future expansion. Be sure to set to 0.) | R | 0 | | |
| 1 | SVP | In SV mode (when PSW.UM = 0), this bit is used to specify whether to restrict access according to the SX, SW, and SR bits of the MPAT register for each protection area.*¹ 0 : As usual, implicitly enable all access in SV mode. 1 : Restrict access according to the SX, SW, and SR bits even in SV mode.*² | R/W | 0 | | |
| 0 | MPE | This bit is used to specify whether to enable or disable MPU function. 0 : Disable 1 : Enable | R/W | 0 | | |
| | Note 1. | When the SVP bit is set to 1, access is restricted according to the setting of eac SV mode. Therefore, specify protection areas before setting the SVP bit to prev program itself from being restricted. | ch protect vent the a | tion area even in ccess of the | | |
| | Note 2. | If access is restricted in SV mode, execution of MDP exceptions or the MIP exception handling itself might not be possible depending on the settings. Be careful to specify settings so that access to the memory area necessary for the exception handler and exception handling is permitted. | | | | |

Table 3.42 MPM Register Contents

(2) MPRC — MPU region control

Bits used to perform special memory protection function operations are located in this register.



| Table 3.43 | MPRC Register Contents |
|------------|------------------------|
|------------|------------------------|

| Bit | Name | Description | R/W | Reset |
|----------|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-------|
| 31 to 16 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 15 to 0 | E15 to E0 | These are the enable bits for each protection area. Bit En is a copy of bit MPAT <i>n</i> .E (where n = 15 to 0). For the number of protection areas, see the hardware manual of the product used. | R/W | 0 |

(3) MPBRGN — MPU base region number

This register indicates the minimum usable MPU area number.



Table 3.44 MPBRGN Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------|------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 5 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | MPBRGN | These bits indicate the smallest number of an MPU area. These bits always indicate 0. | R | 0 |

(4) MPTRGN — MPU end region number

This register indicates the maximum usable MPU area number + 1.



| ble 3.45 MPTRGN Register Conten |
|---------------------------------|
|---------------------------------|

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 5 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 4 to 0 | MPTRGN | These bits indicate the largest number of an MPU area + 1. These bits indicate the maximum number of MPU areas incorporated into the hardware. | R | *1 |

Note 1. For details, see the hardware manual of the product used.

Ta
(5) MCA — Memory protection setting check address

This register is used to specify the base address of the area for which a memory protection setting check is to be performed.



Table 3.46 MCA Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|------------------|--------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | MCA31 to MCA0 | These bits are used to specify the starting address of the memory area which subjects to a memory protection setting check in bytes. | R/W | Undefined |

(6) MCS — Memory protection setting check size

This register is used to specify the size of the area for which a memory protection setting check is to be performed.



Table 3.47 MCA Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | MCS31 to MCS0 | These bits are used to specify the size of the memory area which subjects to a memory protection setting check and the size of the target area in bytes. Because the specified size is assumed to represent an unsigned integer, it is not possible to check an area in the direction in which the address value decreases relative to the MCA register value. Do not specify 0000 0000 _H for the MCS register. | R/W | Undefined |

(7) MCC — Memory protection setting check command

This command register is used to start a memory protection setting check.



Table 3.48 MCC Register Contents

| Bit | Name | Description | R/W | Value after Reset |
|---------|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | MCC31 to MCC0 | When any value is written to the MCC register, a memory protection setting check starts. By setting up the MCA / MCS register and then writing to the MCC register, results are stored in MCR. Because the check is started by any written value, a check can be started by using r0 as the source register without using any unnecessary registers. Note that, for the check, the results are applied according to each area setting regardless of the state of the PSW.UM bit. When the MCC register is read, value 0000 0000 _H is always returned. | R/W | 0 |

RENESAS

(8) MCR — Memory protection setting check result

This register is used to store the results of a memory protection setting check.

Be sure to clear bits 31 to 9, 7, and 6.

CAUTIONS

- 1. If the specified area to be checked crosses $0000\ 0000_{H}$ or 7FFF FFFF_H, it is judged as an area setting error, and the MCR.OV bit is set to 1. This means that the MCR.OV bit must be checked to access the check results. Do not use the check result until it is confirmed that the result is not invalid (OV = 0).
- 2. When the default set (MPM.DX, DW, DR) is set to 1, it disables sometimes to get the correct result. If enabling the specified default operation, do not use the memory protection setting check function.

| | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
|-----|----|----|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|--------------------------------|
| MCR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | | | | | | | | | 1 |
| _ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | _ |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | OV | 0 | 0 | SXE | SWE | SRE | UXE | UWE | URE | Value after reset Undefined |

| Bit | Name | Description | R/W | Value after Reset |
|---------|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 9 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 8 | OV | If the specified area includes $0000\ 0000_H$ or 7FFF FFFF _H , 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 7, 6 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5 | SXE | If the specified area is contained within one protection area and execution is R permitted for that area in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | | Undefined |
| 4 | SWE | If the specified area is contained within one protection area and writing to that area is permitted in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | | Undefined |
| 3 | SRE | If the specified area is contained within one protection area and reading from F that area is permitted in supervisor mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | | Undefined |
| 2 | UXE | If the specified area is contained within one protection area and execution is R permitted for that area in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | | Undefined |
| 1 | UWE | If the specified area is contained within one protection area and writing to that area is permitted in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |
| 0 | URE | If the specified area is contained within one protection area and reading from that area is permitted in user mode, 1 is stored in this bit. In other cases, 0 is stored in this bit. | R/W | Undefined |

Table 3.49 MCC Register Contents



(9) MPLAn — Protection area minimum address

These registers indicate the minimum address of area n (where n = 0 to 15). The number of protection area n depends on the hardware specifications. For details, see the hardware manual of the product used.



| Table 2 EO | Deviator | Contonto |
|------------|----------|----------|
| Table 3.50 | Register | Contents |

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 2 | MPLA31 to MPLA2 | These bits indicate the minimum address of area n. The MPLAn.MPLA1-0 bits are used implicitly set to 0. | R/W | Undefined |
| 1, 0 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |

(10) MPUAn — Protection area maximum address

These registers indicate the maximum address of area n (where n = 0 to 15). The number of protection area n depends on the hardware specifications. For details, see the hardware manual of the product used.



| Table 3 51 | MPII ^Δ n | Register | Contents |
|-------------|----------------------------|----------|----------|
| Iable J.J I | INF UAI | Register | Contents |

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------------------|------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 2 | MPUA31 to MPUA2 | These bits indicate the maximum address of area n. The MPUAn.MPUA1-0 bits are used implicitly set to 1. | R/W | Undefined |
| 1, 0 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |



These registers indicate the attributes of area n (where n = 0 to 15). The number of protection area n depends on the hardware specifications. For details, see the hardware manual of the product used.



| Bit | Name | Description | R/W | Value after Reset |
|----------|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 26 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 25 to 16 | ASID | These bits indicate the ASID value to be used as the area match condition. | R/W | Undefined |
| 15 to 8 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 7 | E | This bit indicates whether area n is enabled or disabled. R 0: Area n is disabled. 1: Area n is enabled. | | 0 |
| 6 | G | 0: ASID match is used as the condition. 1: ASID match is not used as the condition. If this bit is 0, MPATn.ASID = ASID.ASID is used as the area match condition. If this bit is 1, the values of MPATn.ASID and ASID.ASID are not used as the area match condition. | | Undefined |
| 5 | SX | This bit indicates the execution privilege for the supervisor mode. *1 0: Execution is disabled. 1: Execution is enabled. | | Undefined |
| 4 | SW | This bit indicates the write permission for the supervisor mode. *1 F 0: Writing is disabled. 1: Writing is enabled. | | Undefined |
| 3 | SR | This bit indicates the read permission for the supervisor mode. ^{*1} R 0: Reading is disabled. 1: Reading is enabled. | | Undefined |
| 2 | UX | This bit indicates the execution privilege for the user mode. 0: Execution is disabled. 1: Execution is enabled. | R/W | Undefined |
| 1 | UW | This bit indicates the write permission for the user mode. F 0: Writing is disabled. F 1: Writing is enabled. F | | Undefined |
| 0 | UR | This bit indicates the read permission for the user mode. 0: Reading is disabled. 1: Reading is enabled | R/W | Undefined |

Table 3.52 MPATn Register Contents

Note 1. If access is restricted in SV mode, execution of MDP exceptions or the MIP exception handling itself might not be possible depending on the settings. Be careful to specify settings so that access to the memory area necessary for the exception handler and exception handling is permitted.



3.6 Cache Operation Function Registers

3.6.1 Cache Control Function System Registers

Cache control function system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID.

| Register No. (regID, seIID) | Symbol | Function | Access Permission |
|--------------------------------|--------|----------------------------------|-------------------|
| SR16, 4 | ICTAGL | Instruction cache tag Lo access | SV |
| SR17, 4 | ICTAGH | Instruction cache tag Hi access | SV |
| SR18, 4 | ICDATL | Instruction cache data Lo access | SV |
| SR19, 4 | ICDATH | Instruction cache data Hi access | SV |
| SR24, 4 | ICCTRL | Instruction cache control | SV |
| SR26, 4 | ICCFG | Instruction cache configuration | SV |
| SR28, 4 | ICERR | Instruction cache error | SV |

 Table 3.53
 Cache Control System Registers



(1) ICTAGL — Instruction cache tag Lo access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of CIST, values that are stored to the tag RAM for the instruction cache are stored. During execution of CILD, values read from the tag RAM for the instruction cache are stored.



| Bit | Name | Description | R/W | Value after Reset |
|----------|------|---------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 11 | LPN | These bits store physical page number bits 24 to 11. Be sure to set bits 31 to 25, and 0. | R/W | Undefined |
| 10 to 6 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 5, 4 | LRU | These bits indicate LRU information of specified cache line. LRU information cannot be freely changed to any value by the CIST instruction. | R/W | Undefined |
| 3 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 2 | L | This bit stores the lock information. | R/W | Undefined |
| 1 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 0 | V | This bit stores valid/invalid information of specified cache line. | R/W | Undefined |

Table 3.54 ICTAGL Register Contents



(2) ICTAGH — Instruction cache tag Hi access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of CIST, values that are stored to the tag RAM for the instruction cache are stored. During execution of CILD, values read from the tag RAM for the instruction cache are stored.



| Bit | Name | Description | R/W | Value after Reset |
|----------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 | WD | When this bit is set to 1 during CIST execution, data RAM of cache is updated. | R/W | Undefined |
| 30 | PD | When this bit is set to 1 during CIST execution, values in the DATAECC field are overwritten to ECC for data RAM. When this value is 0, ECC is generated automatically from the write data. | R/W | Undefined |
| 29 | WT | When this bit is set to 1 during CIST execution, tag RAM of cache is updated. | R/W | Undefined |
| 28 | PT | When this bit is set to 1 during CIST execution, values in the TAGECC field are overwritten to ECC for tag RAM. When this value is 0, ECC is generated automatically from the write data. | R/W | Undefined |
| 27 to 24 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 23 to 16 | DATAECC | These bits store ECC for data RAM. | R/W | Undefined |
| 15 to 8 | TAGECC | These bits store ECC for tag RAM. Write 0 to bits 15 and 14. | R/W | Undefined |
| 7 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 6 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | Undefined |
| 5 to 2 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 1, 0 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | Undefined |

Table 3.55 ICTAGH Register Contents

(3) ICDATL — Instruction cache data Lo access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of CIST, values that are stored to the tag RAM for the instruction cache are stored. During execution of CILD, values read from the tag RAM for the instruction cache are stored.



| Table 3.56 | ICDATL | Register | Contents |
|------------|--------|------------|------------|
| 10010 0100 | | i togiotoi | 0011101110 |

| Bit | Name | Description | R/W | Value after Reset |
|---------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 0 | DATAL | Bits 31 to 0, 95 to 64, 159 to 128, or 223 to 192 are stored among the instruction data of a block in the specified cache line. The stored bits are specified by the offset of index. Offset of index = 00000: Bits 31 to 0 Offset of index = 01000: Bits 95 to 64 Offset of index = 10000: Bits 159 to 128 Offset of index = 11000: Bits 223 to 192 | R/W | Undefined |



Value after

(4) ICDATH — Instruction cache data Hi access

This register is used by the CIST/CILD instruction in relation to the instruction cache. During execution of CIST, values that are stored to the tag RAM for the instruction cache are stored. During execution of CILD, values read from the tag RAM for the instruction cache are stored.



| Table 3 | 57 | ICDATH Register Contents |
|---------|-----|--------------------------|
| Name | Des | cription |

| Bit | Name | Description | R/W | Reset |
|---------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----------|
| 31 to 0 | DATAH | Bits 63 to 32, 127 to 96, 191 to 160, or 255 to 224 are stored among the instruction data of a block in the specified cache line. The stored bits are specified by the offset of index. Offset of index = 00000: Bits 63 to 32 Offset of index = 01000: Bits 127 to 96 Offset of index = 10000: Bits 191 to 160 Offset of index = 11000: Bits 255 to 224 | R/W | Undefined |

(5) ICCTRL — Instruction cache control

This register is used to control the instruction cache.



| Bit | Name | Description | R/W | Value after Reset |
|----------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 17 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 16 | _ | (Reserved for future expansion. Be sure to set to 1.) | R | 1 |
| 15 to 9 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 8 | ICHCLR | When this bit is set to 1, the entire instruction cache is cleared. After this bit is set to 1, it is read as 1 until clearing is completed. The bit is cleared to 0 once clearing is completed. | R/W | 0 |
| 7 to 3 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 2 | ICHEIV | When this bit is set to 1, the instruction cache is automatically set as invalid (the ICHEN bit is cleared to 0) whenever a cache error occurs. | R/W | 0 |
| 1 | ICHEMK | When this bit is set to 1, it masks notification of cache error exceptions for the CPU after a cache error has occurred. | R/W | 1 |
| 0 | ICHEN | This bit indicates valid/invalid status of instruction cache. 0: Instruction cache is invalid 1: Instruction cache is valid This bit is read as the previous value until the setting is actually reflected in the instruction cache. | R/W | 1 |

Table 3.58 ICCTRL Register Contents



(6) ICCFG — Instruction cache configuration

This register indicates the instruction cache configuration.



Table 3.59 ICCFG Register Contents

| Name | Description | R/W | Value after Reset |
|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| ICHSIZE | These bits indicate the size (in Kbytes) of the instruction cache. 000 0000: No instruction cache 000 0001: 1 Kbyte 000 0010: 2 Kbytes 000 0100: 4 Kbytes 000 1000: 8 Kbytes 001 0000: 16 Kbytes 010 0000: 32 Kbytes 100 0000: 64 Kbytes Other than above: Setting prohibited | R | *1 |
| ICHLINE | These bits indicate the number of lines for each way in the instruction cache. 0000: No instruction cache 0001: 32 lines 0010: 64 lines 0100: 128 lines 1000: 256 lines Other than above: Setting prohibited | R | *1 |
| ICHWAY | These bits indicate the number of ways in the instruction cache. 0000: No instruction cache 0001: 1 way 0010: 2 ways 0100: 4 ways 1000: 8 ways Other than above: Setting prohibited | R | *1 |
| ICHW4 | AY to 1 | AY These bits indicate the number of ways in the instruction cache. 0000: No instruction cache 0001: 1 way 0010: 2 ways 0100: 4 ways 1000: 8 ways Other than above: Setting prohibited to 1. The volue ofter recet depende on the bardware exception tions. For details, see the | AY These bits indicate the number of ways in the instruction cache. R 0000: No instruction cache 0001: 1 way 0010: 2 ways 0100: 4 ways 1000: 8 ways Other than above: Setting prohibited to 1. The value ofter reset depends on the bardware essetilizations. For details, see the bardware |

Note 1. The value after reset depends on the hardware specifications. For details, see the hardware manual of the product used.



This register is used to store cache error information for the instruction cache.

After the ICHERR bit is set to 1, any subsequent cache error information that is generated is not stored until this setting is explicitly cleared to 0.



| 31CISTWThis bit is set to indicate that the destination way specified for a CISTI instruction was in error. Although the entry information is overwritten so that writing is completed, the V bit will be cleared the next time the cache line is read (i.e. reading will be judged to have missed the cache). However, setting of this bit is not accompanied by an exception for the CPU.R/W030—(Reserved for future expansion. Be sure to set to 0.)R029ESMHError status: Multi hitR/WU28ESPBSEError status: WAY errorR/WU27ESTE1Error atatus: Tag DAM 4 bit errorR/WU | Reset |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| 30—(Reserved for future expansion. Be sure to set to 0.)R029ESMHError status: Multi hitR/WU28ESPBSEError status: WAY errorR/WU27ESTE1Error status: Too DAM 4 bit errorD/M 4 bit error |) |
| 29 ESMH Error status: Multi hit R/W U 28 ESPBSE Error status: WAY error R/W U 27 ESTE1 Error status: Too BAM 4 bit error BAM 4 |) |
| 28 ESPBSE Error status: WAY error R/W U 27 ESTED Error status: Too DAM 4 bit error DAM 4 bit error | Indefined |
| 27 ESTE1 Error atotuo: Tog DAM 1 bit error DAM | Indefined |
| 27 ESTET Error status: Tag RAM 1-bit error R/W U | Indefined |
| 26 ESTE2 Error status: Tag RAM 2-bit error R/W U | Indefined |
| 25 ESDC Error status: Data RAM 1-bit correction R/W U | Indefined |
| 24 ESDE Error status: Data RAM 2-bit error R/W U | Indefined |
| 23, 22 — (Reserved for future expansion. Be sure to set to 0.) R 0 |) |
| 21 ERMMH Error exception notification mask : Multi bit R/W 0 |) |
| 20 ERMPBSE Error exception notification mask : WAY error R/W 0 | |
| 19 ERMTE1 Error exception notification mask : Tag RAM 1-bit error R/W 0 |) |
| 18 ERMTE2 Error exception notification mask : Tag RAM 2-bit error R/W 0 |) |
| 17 ERMDC Error exception notification mask : Data RAM 1-bit correction R/W 0 | |
| 16 ERMDE Error exception notification mask : Data RAM 2-bit error R/W 0 |) |
| 15 — (Reserved for future expansion. Be sure to set to 0.) R 0 | |
| 14, 13 ICHEWY These bits retain the way number where a cache error occurred. R/W U | Indefined |
| 12 to 5 ICHEIX These bits retain the cache index where a cache error occurred. R/W U | Indefined |
| 4 ICHERQ When this bit is set to 1, this bit indicates that cache error exception R/W 0 notification is in progress. However, if cache error exception notification has been masked, the CPU is not notified even when 1 has been set to this bit. |) |
| 3 ICHED This bit indicates that an error has occurred in data RAM. R/W 0 |) |
| 2 ICHET This bit indicates that an error has occurred in tag RAM. R/W 0 |) |
| 1 — (Reserved for future expansion. Be sure to set to 0.) R 0 |) |
| 0 ICHERR This bit is set to 1 when a cache error has occurred. R/W 0 |) |

Table 3.60 **ICERR Register Contents**



3.7 Data Buffer Operation Registers

3.7.1 Data Buffer Control System Registers

Data buffer control system registers are read from or written to by using the LDSR and STSR instructions and specifying the system register number, which is made up of a register number and selection ID. For data buffer functions, see the hardware manual of the product used.

 Table 3.61
 List of Data Buffer Operation Registers

| Register No. (regID, selID) | Symbol | Function | Access Permission |
|--------------------------------|--------|------------------------------|-------------------|
| SR 24, 13 | CDBCR | Data buffer control register | SV |

(1) CDBCR — Data buffer control register

This is the register for controlling data buffer.



| Table 3.62 CDBCR Register Contents | Table 3.62 | CDBCR | Register | Contents |
|------------------------------------|------------|-------|----------|----------|
|------------------------------------|------------|-------|----------|----------|

| Bit | Name | Description | R/W | Value after Reset |
|---------|--------|-------------------------------------------------------------------------------------------------------------------------|-----|----------------------|
| 31 to 2 | _ | (Reserved for future expansion. Be sure to set to 0.) | R | 0 |
| 1 | CDBCLR | When this bit is set to 1, data buffer is all cleared. This bit is always read as 0. | W | 0 |
| 0 | CDBEN | This bit specifies enables or disables of the data buffer. 0: Data buffer is disabled. 1: Data buffer is enabled. | R/W | 1 |



Section 4 EXCEPTIONS AND INTERRUPTS

An exception is an unusual event that forces a branch operation from the current program to another program, due to certain causes.

A program at the branch destination of each exception is called an "exception handler".

CAUTION

This CPU handles interrupts as types of exceptions.

4.1 Outline of Exceptions

This section describes the elements that assign properties to exceptions, and shows how exceptions work.

4.1.1 Exception Cause List



| | | EBV | 0 | ω | ω | ω | ω | S | S | S | v | v | ω | ω | ω | S |
|--------------|-----------------------------|----------------------------------------|---------------------------|--------------------------------------|-------------------------------------------------|---------------------------------------|-----------------------------------------------------|---------------------------------------------------------|-------------------------------------------------|-------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|------------------------------------|------------------------------------------------------|---------------------------------------|--------------------------------------|
| | | £ | 0 | 0 | - | 0 | 0 | - | - | - | ~ | ~ | - | - | - | - |
| | | đ | 0 | ~ | - | - | S | - | - | - | ~ | ~ | - | - | S | s |
| | (PSW) | ₽ | - | - | - | - | - | - | - | - | ~ | ~ | - | - | - | - |
| | Update | M | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | nowledgment dition (PSW) | đ | × | × | × | 0 | 0 | × | × | × | × | × | × | × | × | × |
| | Ack Con | ₽ | × | × | × | × | 0 | × | × | × | × | × | × | × | × | × |
| | Order *2 | Priorit | I | ~ | 5 | ю | *4 | - | с | 4 | ъ | Q | ۷* | ۷* | ۲* | 89 |
| | Priority | Priority Level | - | e | ю | ю | 4 | 10 | 10 | 10 | 0 | 0 | 1 | 1 | 1 | 12 |
| | | Exception Cause Code * ⁵ | None | EOH | 10 _H -1F _H * ³ | F0 _H | 1000 _H -11FF _H * ⁶ | H06 | 10 _Н -1F _H * ³ | 60 _H | 80 _H -82 _H *9 | AOH | COH | 91 _H | 71 _H | 8000 _H -80FF _H |
| | | Return/ Restoration | I | No | No | Yes | Yes | Yes | N | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| 1/2) | | Saved Resource | I | H | H | H | Ш | Ë | H | H | 븬 | 븬 | H | H | Ш | Ξ |
| ause List (| | Type *1 | Terminating | Terminating | Terminating | Terminating | Terminating | Resumable | Resumable | Resumable | Resumable | Resumable | Resumable | Resumable | Resumable | Pending |
| Exception Ca | | Source | Reset input* ³ | Interrupt controllef ³ | System error input ^{*3} | Interrupt controller ^{*3} | Interrupt controller ^{*3} | Memory protection violation | Error input during instruction fetch *3 | Execution of a reserved instruction | Execution of a coprocessor instruction/ access permission violation | Execution of a privileged instruction/ access permission violation | Misaligned access occurrence | Memory protection violation | Execution of an FPU instruction | Execution of the |
| Table 4.1 | | Name | Reset | FENMI interrupt | System error | FEINT interrupt | User interrupt | Memory protection exception (execution privilege) | System error | Reserved instruction exception | Coprocessor unusable exception | Privilege instruction exception | Misalignment exception | Memory protection exception (access privilege) | Floating-point operation exception | System call |
| | | Exception | RESET | FENMI | SYSERR | FEINT | EIINT0-511 | MIP | SYSERR | RIE | ИСРОР | PIE | MAE | MDP | FPINT | SYSCALL |

RH850G3MH Software

R01US0143EJ0100 Rev.1.00 Mar 05, 2015 RENESAS

| Table 4.1 Exception Cause List (2/2) | Acknowledgment Priority Order * ² Condition (PSW) Update (PSW) | Saved Return/ Exception Priority Source Type *1 Resource Restoration Cause Code *5 Level Priority ID NP UM ID NP EP EBV | el trap Execution of the Pending FE Yes 31 _H -3F _H 12 * ⁸ x x 0 1 1 1 s FETRAP instruction | l trap 0 Execution of the Pending EI Yes 40 _H -4F _H 12 * ⁸ x x 0 1 s 1 s TRAP instruction | I trap 1 Execution of the Pending EI Yes 50 _H -5F _H 12 * ⁸ x x 0 1 s 1 s TRAP TRAP instruction | Vote: s: Retained, x: Not an acknowledgment condition | Note 1. For details, see Section 4.1.3, Types of Exceptions. | Note 2. The acknowledgment priority for exceptions is checked by the priority level, and then priority. A smaller value has a higher priority. For details, see Section 4.1.4, Exception Acknowledgment Conditions and Priority Order. | Note 3. For details, see the hardware manual of the product used. | Note 4. The priorities of EIINT0 to EIINT511 vary depending on the register setting. For details, see Section 4.1.5, Interrupt Exception Priority and Priority Masking . | Note 5. The lower 16 bits of the exception cause code are shown. The higher 16 bits of the exception cause code contain the detailed code defined for each exception. These bits are 0000 _H unless otherwise specified in the description of the function. | Vote 6. 1000 _H to 11FF _H (channels 0 to 511) are selected according to the channel. | Vote 7. This depends on the operation order of instructions. | Note 8. These exceptions occur exclusively because they occur due to instruction execution. There is no priority within the same priority level. | Vote 9. 80 _H to 82 _H correspond to the coprocessor use permission (CU0 to CU2), respectively. |
|--------------------------------------|------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| nble 4.1 E | | Ŏ | ir FL | ар о о | тар 1- 1- 1- 1- 1- 1- 1- 1- 1- 1- 1- 1- 1- | ite: s: Retainec | ite 1. For deta | te 2. The ackr For detai | ite 3. For deta | ote 4. The prio For detai | te 5. The low These bi | ite 6. 1000 _H tc | ite 7. This dep | ite 8. These ex | ite 9. 80 _H to 8. |
| Та | | eption Name | RAP FE level | APO EI level ti | AP1 EI level ti | No | No | NG | No | NG | NG | No | No | No | No |

RENESAS

4.1.2 Overview of Exception Causes

The following is an overview of the exception causes handled in this CPU.

(1) RESET

These are signals generated when inputting a reset. For details, see Section 8, RESET.

(2) FENMI, FEINT, and EIINT

These are interrupt signals that are input from the interrupt controller to activate a certain program. For details about the interrupt functions, see **Section 3.3, Interrupt Function Registers** and the specifications of the interrupt controller incorporated in your product.

(3) SYSERR

This is a system error exception. This exception occurs when an error defined by the hardware specifications is detected. An error that occurs at an instruction fetch access is reported as a resumable-type SYSERR exception. Other errors are reported as a terminating-type SYSERR exception.

CAUTION

The cause of an SYSERR exception is determined according to the hardware functions. For details, see the hardware manual of the product used.

(4) FPINT

These are exceptions that occur when a floating-point instruction is being executed. For details, see Section 6.1, Floating-Point Operation.

(5) MIP and MDP

These are exceptions that occur when the MPU detects a violation. Detecting an exception is performed when the address at which the instruction will access the memory is calculated. For details, see Section 5.1, Memory Protection Unit (MPU).

(6) RIE

This is a reserved instruction exception. This exception occurs when an attempt is made to execute the opcode of an instruction other than an instruction whose operation is defined. The operation is the same as a RIE instruction whose operation is defined. For details, see **7.1.3**, **Reserved Instructions** in **Section 7**, **INSTRUCTION**.

(7) PIE

This is a privilege instruction exception. This exception occurs when an attempt is made to execute an instruction that does not have the required privilege. For details, see Section 2.1.3, CPU Operating Modes and Privileges, Section 2.2, Instruction Execution, and Section 2.5.3(1) LDSR and STSR.

(8) UCPOP

This is an exception that occurs when an attempt is made to execute a coprocessor instruction when the coprocessor in question is not usable. For details, see **Section 2.4, Coprocessors**.

(9) MAE

This is an exception that occurs when the result of address calculation is a misaligned address. For details, see **Section 2.6.3, Data Alignment**.

(10) TRAP, FETRAP, and SYSCALL

These are exceptions that occur according to the result of instruction execution. For details, see **Section 7, INSTRUCTION**.

4.1.3 Types of Exceptions

This CPU divides exceptions into the following three types according how they are executed.

- Terminating exceptions
- Resumable exceptions
- Pending exceptions

(1) Terminating exceptions

In the case of an exception of this type, the exception is acknowledged by interrupting the current instruction before its operation is executed. These exceptions include interrupts and exceptions that are generated by sources that are unrelated to the program currently running, such as hardware errors.

These interrupts do not occur as a result of executing the current instruction and are not related to the instruction. When an interrupt occurs, the PSW.EP bit is cleared to 0, unlike other exceptions. Consequently, termination of the exception handler routine is reported to the external interrupt controller when the return instruction is executed. Be sure to execute an instruction that returns execution from an interrupt while the PSW.EP bit is cleared to 0.

CAUTION

The PSW.EP bit is cleared to 0 only when an interrupt (INT0 to INT511, FEINT, or FENMI) is acknowledged. It is set to 1 when any other exception occurs.

If an instruction to return execution from the exception handler routine that has been started by generation of an interrupt is executed while the PSW.EP bit is set to 1, the resources on the external interrupt controller might not be released, causing malfunctioning.

The return PC of a terminating exception is the PC of the terminated instruction (current PC).

(2) Resumable exceptions

This is an exception acknowledged during the execution of instruction operation before the execution is finished. The floating-point operation exception is an example of an exception of this type. Generalpurpose registers and system registers are not updated due to the occurrence of an exception of this type. The PC value on return from the exception continues to point to the instruction where the exception occurred, so execution can be restarted from the state of before the exception occurred.

The return PC of a resumable exception is the PC of the instruction which caused the exception (current PC).



(3) Pending exceptions

This is an exception acknowledged after the execution of an instruction finishes as a result of executing the instruction operation. Pending exceptions include software exceptions. Because pending exceptions occur as a result of normal instruction execution, the processing resumes with the instruction following the instruction that caused the pending exceptions when processing control is returned. The original processing can be normally continued after the exception handling.

The return PC of a pending exception is the PC of the next instruction (next PC).

4.1.4 Exception Acknowledgment Conditions and Priority Order

The CPU acknowledges only one exception at specific timing based on the exception acknowledgment conditions and priority order. The exception to be acknowledged is determined based on the exception acknowledgment conditions and priority order, as shown in **Figure 4.1** below.



Figure 4.1 Exception Acknowledgment Conditions and Priority Order

In **Table 4.1**, an exception with "0" in the acknowledgment condition column can be acknowledged when the corresponding bit is "0". For this kind of exception, acknowledgment is held pending when the corresponding bit is "1". When it changes to "0" and the acknowledgment conditions are met, acknowledgment of the exception becomes possible. If no value is specified for a bit, it is not an acknowledgment condition. If multiple bits are specified as conditions, all the conditions must be met simultaneously.

If more than two exceptions satisfy the acknowledgment conditions simultaneously, one exception is selected according to the priority order. The priority order is determined in multiple stages; priority level, and then priority. A smaller number has a higher priority.

When a terminating exception is not acknowledged, it is held pending. If it occurs at the time of a reset, it is not held pending. For details, see **Section 4.2.1**, **Special Operations**.

For details about acknowledgment conditions, priority level, and priority, see **Table 4.1, Exception Cause List**.

4.1.5 Interrupt Exception Priority and Priority Masking

An interrupt (EIINT*n*) can be masked for each exception priority or interrupt priority by setting registers. This function allows the software implementation of an interrupt ceiling with a more flexible software structure and no maintenance.

Figure 4.2 shows an overview of the functions of interrupt exception priority and priority masking.



Figure 4.2 Interrupt Exception Priority and Priority Masking



(1) Interrupt priority

For an interrupt (EIINT*n*), the exception priority can be changed by setting registers.

(2) Interrupt priority mask

EIINT*n* might be masked at different priorities by the ISPR register and PMR register. These registers should be used as follows.

For the ISPR register, the bit corresponding to the priority is set (1) when the hardware acknowledges an interrupt, and interrupts with a lower priority are masked. When the EIRET instruction corresponding to the interrupt is executed, the corresponding bit of the ISPR register is cleared (0) to clear the mask.

This automatic interrupt ceiling makes multiplexed interrupt servicing easy without using software control.

The PMR register allows you to mask specific interrupt priorities with software. Use it to raise the level of the interrupt ceiling temporarily in a program. The mask setting specified by the ISPR register and the mask setting of PMR might overlap, and an interrupt is masked if it is masked with one or the other of them. Normally, use the PMR register to raise the ceiling value from the ceiling value of the ISPR register.

The function of the INTCFG register allows you to disable auto update of the ISPR register upon acknowledgment of and return from an interrupt. To perform interrupt ceiling control by using software without using the function of the ISPR register, set (1) the ISPC bit of the INTCFG register, clear the ISPR register, and then control the ceiling value with software by using the PMR register.

Also, when you are using the PMR register, you can check if any interrupt is masked with the PMR register by using the ICSR register.

4.1.6 Return and Restoration

When exception handling has been performed, it might affect the original program that was interrupted by the acknowledged exception. This effect is indicated from two perspectives: "Return" and "Restoration".

- Return : Indicates whether or not the original program can be re-executed from where it was interrupted.
- Restoration : Indicates whether or not the processor statuses (status of processor resources such as general-purpose registers and system registers) can be restored as they were when the original program was interrupted.

An exception that cannot be returned or restored from ("No" in **Table 4.1**) might cause the return PC to be lost, making it impossible to return from the exception to the original processing by using a return instruction. An exception whose trigger cannot be selected is an unreturnable or unrestorable exception.

For an unrestorable exception, it is possible to return to the original program flow. However, because the state before the occurrence of the exception cannot be restored at that point, care must be taken in continuing subsequent program operation.



4.1.7 Context Saving

To save the current program sequence when an exception occurs, appropriately save the following resources according to the function definitions.

- Program counter (PC)
- Program status word (PSW)
- Exception cause code (EIIC, FEIC)
- Work system register (EIWR, FEWR)

The resource to use as the saving destination is determined according to the exception type. Saved resource determination is described below.

(1) Context saving

Exceptions with certain acknowledgment conditions might not be acknowledged at the start of exception handling, based on the pending bits (PSW.ID and NP bits) that are automatically set when another exception is acknowledged.

To enable processing of multiple exceptions of the same level that can be acknowledged again, certain information about the corresponding return registers and exception causes must be saved, such as to a stack. This information that must be saved is called the "context".

In principle, before saving the context, caution is needed to avoid the occurrence of exceptions at the same level.

The work system registers that can be used for work to save the context, and the system registers that must be at least saved to enable multiple exception handling are called basic context registers. These basic context registers are provided for each level.

 Table 4.2
 Basic Context Registers

| Exception Level | Basic Context Registers |
|-----------------|-------------------------|
| El level | EIPC, EIPSW, EIIC, EIWR |
| FE level | FEPC, FEPSW, FEIC, FEWR |



4.2 Operation When Acknowledging an Exception

Check whether each exception that is reported during instruction execution is acknowledged according to the priority. The procedure for exception-specific acknowledgment operation is shown below.

- (1) Check whether the acknowledgment conditions are satisfied and whether exceptions are acknowledged according to their priority.
- (2) Calculate the exception handler address according to the current PSW value^{*1}.
- (3) For FE level exceptions, the following processing is performed.
- Saving the PC to FEPC
- Saving the PSW to FEPSW
- Storing the exception cause code in FEIC
- Updating the PSW^{*2}
- Store the exception handler address calculated in (2) in the PC, and then pass control to the exception handler.
- (4) For EI level exceptions, the following processing is performed.
- Saving the PC to EIPC
- Saving the PSW to EIPSW
- Storing the exception cause code in EIIC
- Updating the PSW^{*2}
- Store the exception handler address calculated in (2) in the PC, and then pass control to the exception handler.
- Note 1. For details, see Section 4.4, Exception Handler Address.
- Note 2. For the values to be updated, see Table 4.1, Exception Cause List.

The following figure shows steps (1) to (4).





Figure 4.3 Operation When Acknowledging an Exception



4.2.1 Special Operations

(1) EP bit of PSW register

If an interrupt is acknowledged, the PSW.EP bit is cleared to 0. If an exception other than an interrupt is acknowledged, the PSW.EP bit is set to 1.

Depending on the EP bit setting, the operation changes when the EIRET or FERET instruction is executed. If the EP bit is cleared to 0, the bit with the highest priority (0 is the highest) among the bits set to 1 in ISPR.ISP15 to ISPR.ISP0 is cleared to 0. Also, the end of the exception handling routine is reported to the external interrupt controller. This function is necessary for correctly controlling resources, such as a request flag, on the interrupt controller when an interrupt is acknowledged or when execution returns from the interrupt.

To return from an interrupt, be sure to execute the return instruction with the EP bit cleared to 0.

(2) Coprocessor unusable exception

For coprocessor unusable exceptions, the exception occurrence opcode corresponding to the status of the CU bit of the PSW register differs according to the specifications of each product.

For coprocessor instructions and defined opcodes, if an attempt is made to execute a coprocessor instruction that is not included in the product or for which the operation state prevents use, or an LDSR or STSR instruction attempts to access a coprocessor system register, a coprocessor unusable exception (UCPOP) immediately occurs.

For details, see Section 2.4.3, Coprocessor Unusable Exceptions.

(3) Reserved instruction exception

If an opcode that is reserved for future function extension and for which no instruction is defined is executed, a reserved instruction exception (RIE) occurs.

However, which of the following two types of operations each opcode is to perform might be defined by the hardware specifications.

- Reserved instruction exception occurs.
- Operates as a defined instruction.

An opcode for which a reserved instruction exception occurs is always defined as an RIE instruction.

(4) Reset

Reset is performed in the same way as exception handling, but it is not regarded as EI level exception or FE level exception. The reset operation is the same that of an exception without acknowledgment conditions, but the value of each register is changed to the value after reset. In addition, execution does not return from the reset status.

All exceptions that have occurred at the same time as CPU initialization are canceled and not acknowledged even after CPU initialization.

For details, see Section 8, RESET.



4.3 Return from Exception Handling

To return from exception handling, execute the return instruction (EIRET or FERET) corresponding to the relevant exception level.

When a context has been saved, such as to a stack, the context must be restored before executing the return instruction. When execution is returned from an irrecoverable exception, the status before the exception occurs in the original program cannot be restored. Consequently, the execution result might differ from that when the exception does not occur.

The EIRET instruction is used to return from EI level exception handling and the FERET instruction is used to return from FE level exception handling.

When the EIRET or FERET instruction is executed, the CPU performs the following processing and then passes control to the return PC address.

- When the EIRET instruction is executed, return PC and PSW are loaded from the EIPC and EIPSW registers.
 When the FERET instruction is executed, return PC and PSW are loaded from the FEPC and FEPSW registers.
- (2) Control is passed to the address indicated by the return PC that were loaded.
- (3) When the EIRET instruction is executed while EP = 0 and INTCFG.ISPC = 0, the CPU updates the ISPR register.

When the FERET instruction is executed, the CPU does not update the ISPR register.

The flow for returning from exception handling using the EIRET or FERET instruction is shown below.





Figure 4.4 Return Instruction-Based Exception Return Flow



4.4 Exception Handler Address

For this CPU, the exception handler address used for execution during reset input, exception acknowledgment, or interrupt acknowledgment can be changed according to the settings.

4.4.1 Resets, Exceptions, and Interrupts

The exception handler address for resets and exceptions is determined by using the direct vector method, in which the reference point of the exception handler address can be changed by using the PSW.EBV bit, RBASE register, and EBASE register. For interrupts, the direct vector method and table reference method can be selected for each channel. If the table reference method is selected, execution can branch to the address indicated by the exception handler table allocated in the memory.

(1) Direct vector method

The CPU uses the result of adding the exception cause offset shown in **Table 4.3**, **Selection of Base Register/Offset Address** to the base address indicated by the RBASE or EBASE register as the exception handler address.

Whether to use the RBASE or EBASE register as the base address is selected according to the PSW.EBV bit^{*1}. If the PSW.EBV bit is set to 1, the EBASE register value is used as the base address. If the bit is cleared to 0, the RBASE register value is used as the base address.

However, reset input and some exceptions^{*2} always refer to the RBASE register.

In addition, user interrupts refer to the RINT bit of the corresponding base register, and reduce the offset address according to the bit status. If the RBASE.RINT bit or EBASE.RINT bit is set to 1, all user interrupts are handled using an offset of $100_{\rm H}$. If the bit is cleared to 0, the offset address is determined according to **Table 4.3**, **Selection of Base Register/Offset Address**.

- Note 1. Exception acknowledgment itself sometimes updates the status of the PSW.EBV bit. In this case, the base register is selected based on the new bit value. For details, see Section 4.4, Exception Handler Address.
- **Note 2.** The exceptions that always reference RBASE are determined according to the hardware specifications.





Figure 4.5 Direct Vector Method



The table below shows how base register selection and offset address reduction function for each exception to determine the exception handler address. The PSW bit value determines the exception handler, based on the value after being updated due to the acknowledgment of an exception.

 Table 4.3
 Selection of Base Register/Offset Address

| | PSW.EBV = 0 | PSW.EBV = 1 | RINT = 0 | RINT = 1 | | | |
|----------------------|-------------|-------------|------------------|------------------|--|--|--|
| | Base | Register | Offset Address | | | | |
| RESET | RBASE | None*1 | 000 _H | 000 _H | | | |
| SYSERR | | EBASE | 010 _H | 010 _H | | | |
| FETRAP | | | 030 _H | 030 _H | | | |
| TRAP0 | | | 040 _H | 040 _H | | | |
| TRAP1 | | | 050 _H | 050 _H | | | |
| RIE | | | 060 _H | 060 _H | | | |
| FPINT | | | 070 _H | 070 _H | | | |
| UCPOP | | | 080 _H | 080 _H | | | |
| MIP/MDP | | | 090 _H | 090 _H | | | |
| PIE | | | 0A0 _H | 0A0 _H | | | |
| Debug* ² | | | 0B0 _H | 0B0 _H | | | |
| MAE | | | 0C0 _H | 0C0 _H | | | |
| (R.F.U.) | | | 0D0 _H | 0D0 _H | | | |
| FENMI | | | 0E0 _H | 0E0 _H | | | |
| FEINT | | | 0F0 _H | 0F0 _H | | | |
| EIINTn (priority 0) | | | 100 _H | 100 _H | | | |
| EIINTn (priority 1) | | | 110 _H | | | | |
| EIINTn (priority 2) | | | 120 _H | | | | |
| EIINTn (priority 3) | | | 130 _H | | | | |
| EIINTn (priority 4) | | | 140 _H | | | | |
| EIINTn (priority 5) | | | 150 _H | | | | |
| EIINTn (priority 6) | | | 160 _H | | | | |
| EIINTn (priority 7) | | | 170 _H | | | | |
| EIINTn (priority 8) | | | 180 _H | | | | |
| EIINTn (priority 9) | | | 190 _H | | | | |
| EIINTn (priority 10) | | | 1A0 _H | | | | |
| EIINTn (priority 11) | | | 1B0 _H | | | | |
| EIINTn (priority 12) | | | 1C0 _H | | | | |
| EIINTn (priority 13) | | | 1D0 _H | | | | |
| EIINTn (priority 14) | | | 1E0 _H | | | | |
| EIINTn (priority 15) | | | 1F0 _H | | | | |

Note 1. An exception generated to update EBV to 0.

Note 2. The exception for debug function.



Base register selection is used to execute the exception handling for resets and some hardware errors by using programs in a relatively reliable area such as ROM instead of areas that are easily affected by soft errors such as RAM and cache areas. The user interrupt offset address reduction function is used to reduce the memory size required by the exception handler for specific system-internal operating modes. The main purpose of this is to minimize the amount of memory consumed in operating modes that use only the minimum functionality, which are used, for example, during system maintenance and diagnosis.

(2) Table reference method

In the direct vector method, there is one user-interrupt exception handler for each interrupt priority level, and interrupt channels that indicate multiple interrupts with the same priority branch to the same interrupt handler, but some users might want to use code areas that differ from the start time for each interrupt handler.

When using the table reference method, if the table reference method is specified as the interrupt channel vector selection method for the interrupt controller, the method for determining the exception handler address when an interrupt request corresponding to that interrupt channel is acknowledged differs as follows.

- (1) In any of the following cases, the exception handler address is determined by using the direct vector method.
- When PSW.EBV = 0 and RBASE.RINT = 1
- When PSW.EBV = 1 and EBASE.RINT = 1
- When the interrupt channel setting is not the table reference method
- (2) In cases other than (1), calculate the table reference position.Exception handler address read position = INTBP register + channel number × 4 bytes
- (3) Read word data starting at the interrupt handler address read position calculated in (2).
- (4) Use the word data read in (3) as the exception handler address.

CAUTION

For details about the interrupt channel settings, see the hardware manual of the product used.



A table of exception handler address read positions corresponding to interrupt channels and an overview of the placement in memory are shown below.

Table 4.4 Exception Handler Address Expansion

| Туре | Exception Handler Address Read Position |
|-----------------------------|-----------------------------------------|
| EIINT interrupt channel 0 | $INTBP + 0 \times 4$ |
| EIINT interrupt channel 1 | INTBP + 1 × 4 |
| | |
| EIINT interrupt channel 510 | INTBP + 510 × 4 |
| EIINT interrupt channel 511 | INTBP + 511 × 4 |



Figure 4.6 Overview of Using the Table Reference Method

For details about the exception handler address selection method settings for each interrupt channel, see the hardware manual of the product used.



4.4.2 System Calls

For system call exceptions, the referenced table entry is selected according to the value of the vector specified based on the opcode and the value of the SCCFG.SIZE bit, and the exception handler address is calculated according to the contents of the table entry and the SCBP register value.

As an example, if table size n is specified by SCCFG.SIZE, the table entry is selected as shown below. Note that if the vector specified by the SYSCALL instruction (vector 8) is greater than table size n, the table entry referenced by vector n + 1 to 255 is table entry 0.

| Table 4.5 | System Calls | | |
|-----------|-----------------------------------------------|------------------------|--|
| Vector | Exception Cause Code | Referenced Table Entry | |
| 0 | 0000 8000 _H | Table entry 0 | |
| 1 | 0000 8001 _H | Table entry 1 | |
| 2 | 0000 8002 _H | Table entry 2 | |
| | | | |
| n – 1 | 0000 8000 _H + (n – 1) _H | Table entry n – 1 | |
| n | 0000 8000 _H + n _H | Table entry n | |
| n + 1 | 0000 8000H + (n + 1) _H | Table entry 0 | |
| | | | |
| 254 | 0000 80FE _H | Table entry 0 | |
| 255 | 0000 80FF _H | Table entry 0 | |

CAUTION

Because table entry 0 is selected even if a vector that exceeds n, which is specified for SCCFG.SIZE, is specified, allocate the error processing routine.



4.4.3 Models for Application

The following describes the relations among the RBASE, EBASE, and PSW.EBV bit, and the models intended for application. Principally, in cases where a reset occurs and there is no main code in the address space, this main code is first expanded into the address space (which is often in DRAM) by bootstrapping to enable execution, or it is used to when inserting an instruction cache into an exception handling routine.

Immediately after a reset, when PSW.EBV = 0, operations use the ROM area where the minimum maintenance code was placed as specified in RBASE. After bootstrapping, and after the required code has been expanded in RAM, the code position in the RAM is set to the EBASE register and the PSB.EBV bit is set to 1^{*1} .

Normally, this is the mode of software operations. As for exceptions or interrupts in the range of normal operations, because they are acknowledged when PSW.EBV = 1, the code operates in the RAM area indicated by EBASE, but in cases where phenomena (such as RAM errors or cache errors) occur that would indicate the RAM code itself has not remained correct, an exception is triggered to clear to 0 the PSB.EBV bit^{*2}. In such cases, there is a possibility that the exception handler itself might not be executed correctly using the code at the position indicated by EBASE, so control is moved to the exception handler in the ROM code indicated by RBASE and the PSW.EBV bit is cleared to 0.

Once the PSW.EBV bit is cleared to 0, even if an ordinary exception were to occur while in this mode, the status of the PSW.EBV bit is handed over, so that a mode enabling correct execution of RAM code is maintained, and operation uses code in the ROM area indicated by RBASE until the PSW.EBV bit is set to 1 by the maintenance code.

- Note 1. Normally, an EIRET or FERET instruction should be used to set the PSW.EBV bit to 1.
- **Note 2.** The hardware specifications determine which exception has which cause, and whether or not an exception is needed to clear PSW.EBV to 0.





Figure 4.7 Example of Model for Application (Operation Flow)



Figure 4.8 Example of Model for Application (Address Map)

Section 5 MEMORY MANAGEMENT

This CPU provides the following functions for managing the memory.

- Memory protection unit (MPU)
- Instruction cache function
- Mutual exclusion function

5.1 Memory Protection Unit (MPU)

Memory protection functions are provided in an MPU (memory protection unit) to maintain a smooth system by detecting and preventing unauthorized use of system resources by unreliable programs, runaway events, etc.

5.1.1 Features

(1) Memory access control

Multiple protection areas can be assigned to the address space. Consequently, unauthorized program execution or data manipulation by user programs can be detected and prevented. The upper and lower limit addresses of each area can be specified so that the address space can be used precisely and efficiently.

(2) Access management for each CPU operation mode

In this CPU, several status bits are used to control access to resources, and these bits are used in combination to perform protection that is appropriate, according to each program's level of reliability.

The initial settings are set as appropriate values in the MPM register. Always use the MPE bit to validate the MPU. The SVP bit should be set to 1 only when protection is also being performed by a supervisor such as an OS.



5.1.2 **Protection Area Settings**

(1) Protection area settings

Set the respective protection areas appropriately. For details about registers, see **Section 3**, **REGISTER SET**.

Some additional description is provided below regarding certain caution points.

(a) E bit

This sets the target protection area setup as enabled or disabled. When disabled, all settings are disabled. Make sure valid setting values have been stored for other protection areas (MPUA, MPLA, and MPAT) at the time when this bit is set to 1.

(b) UX, UR, and UW bits

These bits indicate the access privileges for the target protection area during user mode.

(c) SX, SR, and SW bits

These bits indicate the access privileges for the target protection area during supervisor mode. These bits are valid only when the MPM.SVP bit has been set to 1. If the MPM.SVP bit has been cleared to 0, protection is not performed while in supervisor mode, regardless of the values of the SX, SR, and SW bits, and the entire address space becomes access-enabled.

(d) G bit and ASID field

These are the G (Global) bit and the ASID field for comparison. When the G bit is cleared to 0, the values in the ASID register are compared to those in the MPAT.ASID field, and protection area settings are applied to determine accessibility only when these values match. When the G bit is set to 1, protection area settings are applied regardless of the ASID values.



5.1.3 Caution Points for Protection Area Setup

(1) Crossing protection area boundaries

When the specified protection areas overlap, the access control settings for the overlapping parts differ depending on the MPM.DX, DW, and DR bits. If access to the protection area is disabled by default, access is enabled by priority; if access to the protection area is enabled by default, access is prohibited by priority.

In other words, when access to protection areas is disabled by default and multiple protection areas have been specified, if access is enabled for either of the protection areas, access is judged to be enabled. If access to the protection area is enabled by default and access is prohibited for either of the protection areas, access is judged to be prohibited.

In addition, the bits for MPM.DX, DW, and DR in this CPU are fixed to 0, and default operation is prohibited.

(2) Invalid protection area settings

Protection area settings are invalid in the following case.

• When value set to lower-limit address is larger than value set to upper-limit address

CAUTION

Note, however, that addresses are handled as unsigned integers (0_H to FFFF FFFF_H).

(3) Memory access spanning contiguous areas to which access is enabled by the MPU

Access to load values from or store values in areas that may be under protection by the MPU should be handled entirely in single areas. Even if access is enabled in contiguous areas for access control by the MPU, access spanning the access-control areas is prohibited. In the case of this CPU, memory access spanning areas under access control only possible in response to double-word access by the ld.dw or st.dw instruction. The prepare, dispose, pushsp, and popsps instructions are handled as repeated rounds of word access, so access by these instructions that spans areas for access control by the MPU is permitted.

In the case of prefetching, when a whole instruction spans MPU access-control areas and the MPU is enabling access for the area containing the entry point, memory protection allows the access to fetch the instruction.


5.1.4 Access Control

In this CPU, accesses are controlled appropriately according to the settings specified as of the step described in **Section 5.1.3, Caution Points for Protection Area Setup**. In any of the cases listed below, the CPU ensures logical integrity by limiting actual access, detecting violations before instruction execution is completed, and setting up exceptions.

- When about to execute an instruction that includes opcode, at an address outside the executable range
- When about to execute an instruction that reads from an address outside the read-accessible range
- When about to execute an instruction that writes to an address outside the write-accessible range

The specifics of access control vary depending on the hardware specifications, but all have the following points in common.

- When the access result is a prohibit judgment, it is not reflected in memory or I/O devices.
- When the access result is an enabled judgment, it is reflected in memory or I/O devices.

CAUTIONS

- 1. Even when access is enabled, there might be cases where access is blocked by another function that prohibits it.
- 2. In some cases, access judged to be prohibited may be executed for a memory or I/O device. The cases are as listed below.
 - Reading local RAM
 - Reading of code flash memory by an instruction prefetched from the instruction cache

Since execution in response to exceptions due to instructions that read from the local RAM or execute the results of prefetching and so on is inhibited, such access does not affect the execution of instructions. However, when a debugger is monitoring access to local RAM or code flash memory, it may observe access judged to be prohibited.



5.1.5 Violations and Exceptions

In this CPU, violations are detected during instruction fetch access or operand access according to the protection area settings, and an exception is generated.

- Execution protection violation (during instruction access)
- Data protection violation (during operand access)

(1) Execution protection violation (MIP exception)

This violation is detected when an instruction is executed. An execution protection violation such as this is detected when attempting to execute an instruction that has been placed in a non-executable area within the program area.

When an execution protection violation is detected, an MIP exception always occurs.

(2) Data protection violation (MDP exception)

This violation is detected during data access by an instruction. A data protection violation such as this is detected when a memory access instruction attempts to access data from an access-prohibited part of the data area.

When a data protection violation is detected, an MDP exception always occurs.

(3) Exception cause code and exception address

When an execution protection violation or data protection violation has been detected, the exception cause code is determined as shown in **Table 5.1**. The determined exception cause code is set to the FEIC register.

The MEA register is used to store either the PC of the instruction that detected the execution protection violation or the access address used when the data protection violation occurred. The MEA register is shared in order to prevent simultaneous occurrence of MIP and MDP exceptions. Also, when a data protection violation occurs, the information of the instruction that caused the violation is stored in the MEI register.

| | | | Bit Number and Bit Name | | | | | | | | | |
|-----------|----------------------------------|---|-------------------------|----|-----|----|----|----|----|----|----|-----------------|
| | Operation Mode When Violation | | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 to 0 |
| Exception | Occurred | - | MS | BL | RMW | SX | SW | SR | UX | UW | UR | - |
| MIP | User mode | 0 | 0 | 0 | 0 | _ | _ | _ | _ | _ | _ | 90 _H |
| | Supervisor mode | 0 | 0 | 0 | 0 | _ | _ | _ | _ | _ | _ | 90 _H |
| MDP | User mode | 0 | *5 | *4 | *3 | 0 | 0 | 0 | 0 | *2 | *1 | 91 _H |
| | Supervisor mode | 0 | - | | | 0 | *2 | *1 | 0 | 0 | 0 | 91 _H |

Table 5.1 Exception Cause Code of Memory Protection Violation

Note 1. When a read violation is caused by an instruction that includes a read operation, either the SR or UR bit is set to 1.

Note 2. When a write violation is caused by an instruction that includes a write operation, either the SW or UW bit is set to 1.

- Note 3. This bit is set to 1 when a violation is caused by the SET1, NOT1, CLR1, or CAXI instruction.
- Note 4. This bit is set to 1 when a violation is caused by the PREPARE, DISPOSE, PUSHSP, or POPSP instruction.

Note 5. This bit is set to 1 when the instruction causing the violation performs a misaligned access.

Note: UR : A violation is detected during a read operation in user mode (PSW.UM = 1).

UW : A violation is detected during a write operation in user mode (PSW.UM = 1).

UX : A violation is detected during instruction execution in user mode (PSW.UM = 1).

SR : A violation is detected during a read operation in supervisor mode (PSW.UM = 0).

SW : A violation is detected during a write operation in supervisor mode (PSW.UM = 0).

SX : A violation is detected during instruction execution in supervisor mode (PSW.UM = 0).

RMW : Set to 1 when the instruction causing the violation contains a read-modify-write operation (SET1, NOT1, CLR1, or CAXI). BL : Set to 1 when the instruction causing the violation performs a block transfer (PREPARE, DISPOSE, PUSHSP, or POPSP). MS : Set to 1 when the instruction causing the violation performs a misaligned access.

5.1.6 Memory Protection Setting Check Function

When configuring programs that provide a service for the OS (etc.), this CPU provides a memory protection setting check function to enable implementation of a service protection function that checks in advance whether or not the data area to be used for the requested operations is within an area that is accessible by the source that called the service. The OS can use this function to verify the suitability of parameters set for system services provided by the user. Also, this verification processing can be completed quickly when compared to software-based area setting read and comparison operations.

(1) Procedure

Set the base address (lower limit) of the target address range to the MCA register and the size of the target range to the MCS register, then use the LDSR instruction (r0 specification is recommended) to access the MCC register and execute a check. The results can be read from the MCR register by the STSR instruction.

CAUTIONS

- 1. If the specified area to be checked crosses $0000\ 0000_{H}$, it is judged as an area setting error, and the MCR.OV bit is set to 1. This means that the MCR.OV bit must be checked to access the check results. Do not use the check result until it is confirmed that the result is not invalid (OV = 0).
- 2. If the default operations specified by using the MPM.DX, DW, and DR bits are enabled (1), the correct result might not be able to be obtained. If enabling the specified default operation, do not use the memory protection setting check function.

(2) Sample code

It is assumed that the memory protection setting check function will be used for the following operations.

```
_service_protection:
   ori
           0x1000, r0, r12
           ADDRESS, r10
                           // Store the start address of the area to be checked to r10
   mov
  mov
          SIZE, r11
                           // Store the size of the area to be checked to r11
   di
          r10, sr8, 5
   ldsr
                           // Set the address to MCA
   ldsr
          r11, sr9, 5
                          // Set the size to MCS
   ldsr
          r0, sr10, 5
                           // Start checking with MCC
          sr11, r12, 5
                           // Get the results from MCR
   stsr
   ei
          0x0100, r12, r0
   andi
   be
          _overflow
                           // Processing of invalid input when OV = 1
  br
           _result_check // Otherwise, result is determined
```



5.2 Cache

For information regarding the specific functions of mounted cache memory and which functions are mounted, see the hardware manual of the product used.

5.2.1 Cache Operation Registers

Figure 5.1 shows the system registers for cache operation. These registers are native contexts, so the hypervisor privilege is required for operation.



Figure 5.1 Cache Operation Registers

5.2.2 Change Cache Use Mode

(1) Change use mode of instruction cache

The instruction cache use mode can be changed by using the ICCTRL.ICHEN bit. To enable an instruction cache, set the ICHEN bit to 1.

To disable the instruction cache, clear the ICHEN bit to 0.

Completion in executing the LDSR instruction that sets ICHEN might not coincide with completion of the instruction cache operations. As in the following sample code, the SYNCI instruction is executed after the settings are changed to ensure that the change in instruction cache settings take effect.

| LDSR | r10, sr24, 4 | 11 | Change the instruction cache settings (ICCTRL) |
|-------|--------------|----|-------------------------------------------------|
| | | | (setting value is stored in r10) |
| SYNCI | | // | Syncs refetch to completion of LDSR instruction |



5.2.3 Cache Operations using CACHE Instruction

The CACHE instruction manipulates data in the specified cache memory.

Such data manipulation by the CACHE instruction starts after updating of the cache memory by all preceding memory access has been completed. Consequently, the result of preceding memory access is guaranteed to be the target for operations using the CACHE instruction. Additionally, a suitable synchronization period is needed following execution of the CACHE instruction to ensure that the results are reflected in subsequent instructions.

(1) Specification method for target of CACHE instruction

There are basically two ways to specify the target for operations.

- Directly specify the address to be accessed : In this CPU, this is called the address specification method. In this case, the cache line containing the specified address is subject to operation.
- Directly specify the cache memory's way number and line number : In this CPU, this is called the index specification method. In this case, no hit judgment for the cache is performed, and the operation is performed on the specified cache index. For details about the cache index specification method, see **Section 5.2.5, Cache Index Specification Method**.

(2) Operations performed using the CACHE instruction

The operations performed on the cache memory are broadly divided into the six types described below. Some of these operations might not be supported, depending on the cache memory to be manipulated (instruction, data, etc.). For details about each operation, see **Section 7**, **INSTRUCTION**.

(a) Cache Hit Block Invalidate / Cache Indexed Block Invalidate (CHBI / CIBI)

This disables the specified cache line. When using the address specification method, the cache line is disabled only when there is a hit. When using the index method, the cache line is disabled. If the specified cache line is locked, it is unlocked. This operation can be used in cases such as when the entire memory cache is initialized by software.

(b) Cache Fetch And Lock (CFAL)

This stores the data at the specified address to the cache memory. At this time, the cache line where the data is stored is locked. This prevents the cache line from being switched. If the target cache line has already been stored in the cache memory, it is simply locked. If the target cache line has already been stored in the cache memory and is not locked, this operation is not performed.

This operation can be used to improve execution efficiency by reducing variations in instruction execution time that occur due to cache misses in the specified memory area.

CAUTION

The target cache line might not be able to be locked, such as when all cache ways are locked. This operation can be used to efficiently monopolize the cache memory, so note with caution the cache locking specifications and the number of cache ways. For details, see the hardware manual of the product used.



(c) Cache Indexed Load / Cache Indexed Store (CILD / CIST)

This operation is used to directly access the cache memory. Values can be written and read, via a system register, at a position in the cache memory specified by using an index. Because cache data and cache tags can be accessed directly, this operation can be used for purposes such as software debugging.

(d) Other operations

Other special operations related to manipulating the bus and memory, such as deleting links to enable efficient exclusive access, might also be defined as cache operations. For details, see **Section 7**, **INSTRUCTION**.

5.2.4 Cache Operation when the PREF Instruction is Executed

The PREF instruction is provided to realize efficient cache access by advising the CPU that an address is likely to be used in a certain way in the near future. Getting the CPU to prefetch data into the cache memory before use in this way can reduce the read wait time when a cache miss occurs.

Assuming support by compilers and other tools, the PREF instruction can be executed regardless of the CPU operating mode. Execution of the PREF instruction does not cause an exception generated by the MPU, and has no effect on logical operations, just like a NOP instruction.

CAUTION

Because a data read request by the PREF instruction is rather speculative, it might not be executed depending on the cache control policy or system conditions. For details, see the hardware manual of the product used.



5.2.5 Cache Index Specification Method

For a cache instruction that uses the index specification method, explicitly specify the cache memory subject to operation in the format shown in **Figure 5.2**, instead of specifying an address. The bit positions (x, y, z) of each field depend on the size of the cache memory incorporated in the CPU core. Information about the incorporated cache memory and size can be read from the ICCFG register.





CAUTION

The Offset field indicates the byte position within the cache line. This setting is not required (i.e., ignored) in normal index specification operations. For a CILD/CIST operation, it is used to specify a position within the cache line when the ICDAT[HL] register is shorter than the cache line length.

5.2.6 Execution Privilege of the CACHE/PREF Instruction

Because the CACHE instruction directly manipulates the contents of the cache memory, privileges are specified according to the type of operation. When the CACHE instruction is executed without the privilege required for the CACHE operation, a privilege instruction exception (PIE) occurs.

On the other hand, the PREF instruction provides information for speculative execution, so it can be executed in any mode.

The privileges required by the different operations performed by the CACHE instruction are shown below.

(a) Operations allowed with the user privilege

Among address specification method operations, operations without a cache lock (CHBI) can be executed in any operation mode.

(b) Operations requiring the supervisor privilege

Among address specification method operations, operations with a cache lock (CFAL) require the supervisor privilege.

In addition, index specification method operations require the supervisor privilege.



5.2.7 Memory Protection for CACHE and PREF Instructions

When manipulating the cache by specifying an address for the CACHE instruction, it might become the target of memory protection by the MPU. Memory protection is judged based on the operating mode in which the CACHE instruction is executed, and it is handled as a data-side access. It is basically handled as a read access, but operations that damage memory consistency are handled as a write access.

No memory protection judgment is performed when using the index specification method or the PREF instruction.

Table 5.2 shows the correspondence between operations and access privileges.

| Instruction | Target | Address/Index | Instruction Execution Privilege | Access Permission |
|---------------------|-------------|---------------|---------------------------------------|----------------------|
| CHBII | Instruction | Address | UM | Read |
| CIBII | Instruction | Index | SV | _ |
| CFALI | Instruction | Address | SV | Read |
| CISTI | Instruction | Index | SV | _ |
| CILDI | Instruction | Index | SV | — |
| (CLL instruction)*1 | — | — | — | _ |

 Table 5.2
 Relationship Between Cache Operations and Permissions

Note 1. Functions as the CLL instruction. For details, see the description of the CLL instruction in **Section 7**, **INSTRUCTION**.



5.3 Mutual Exclusion

This CPU provides instructions that enable shared resources to be controlled mutually exclusively from multiple programs when the system is operating in a multi-processor environment.

When using mutual exclusion, mutual exclusion variables have to be defined in the memory and all programs must operate in accordance with the appropriate instruction flow.

CAUTION

Embedded CPUs in a single-processor configuration use a programming model in which data coherence is maintained by disabling the acknowledgment of maskable interrupts. This is a very easy and sure method of maintaining data coherence, but naturally in a multi-processor, multiple programs might be executing and attempting to use the data at the same time. In this case it is not possible to maintain data coherence simply by disabling maskable interrupt acknowledgment.

5.3.1 Shared Data that does not Require Mutual Exclusion Processing

This CPU maintains data access coherence even in a multi-processor environment by enabling the following types of access.

- Access in which the data is aligned to the size that matches the data type (aligned access)
 - LD, ST, SLD, SST, LDL, and STC instructions
- Access by using a bit manipulation instruction (SET1, CLR1, or NOT1) (read-modify-write)
- Access by using the CAXI instruction (read-modify-write)

With some exceptions, mutual exclusion is achieved by using these types of data access. In other words, it is guaranteed that while one CPU is executing the instructions that perform the above data accesses, another CPU is not accessing the data in question. This is known as an instruction being executed atomically or an instruction providing an atomic guarantee.

Note that the atomic execution of an instruction means that a data access bus transaction completes with no disruption; it does not necessarily mean that a series of transactions has been completed.

CAUTION

The extent to which coherency is guaranteed might be limited, depending on the hardware specifications. For example, for some memories, coherency might not be preserved even if aligned access is used. For details, see the hardware manual of the product used.



5.3.2 Performing Mutual Exclusion by Using the LDL.W and STC.W Instructions

The LDL.W and STC.W instructions can be used to perform mutual exclusion over multiple data arrays.

When acquiring a lock by using the LDL.W and STC.W instructions in a pair, first a link is created by using the LDL.W instruction and then the STC.W instruction is executed.

At this time, if data is written to the address at which the link was created before the STC.W instruction is executed, the link is immediately deleted, the subsequent execution of the STC.W instruction fails, and a lock fails to be acquired.

(1) Link

Each link (LLbit) includes information on the address at which it was created, which is used to control whether the STC instruction executes successfully or fails, and whether the link is deleted.

A link is created when the LDL.W instruction is executed. If the LDL.W instruction is executed again after a link has been created, another link is created, which overwrites the first link. In other words, only one link exists at a time, and that link contains the address information of the LDL.W executed last.

Links are deleted when certain event or address conditions are satisfied. **Table 5.3** shows the link deletion conditions. A link is deleted if any of the conditions shown in **Table 5.3** are satisfied.

| Target Link | Event Condition | Remark |
|-----------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| All links in the system (including those in other CPU cores) | If a write operation occurs in a 32-byte-aligned address range that includes the address of the link in question | ST, SST, and STC instructions SET1, NOT1, CLR1, and CAXI instructions PREPARE and PUSHSP instructions |
| CPU core link | Execution of STC.W instruction | The link is deleted whether the instruction executes successfully or fails |
| | Execution of CLL instruction | Use a CLL instruction to clear a link in a function explicitly (abortion of an atomic operation). |
| | Exception acknowledgment | |
| | Execution of return instruction | Does not include CTRET instruction |

Table 5.3 Link Deletion Conditions

CAUTION

Links that are deleted by a write operation are deleted in 32-byte units. Therefore, the best way to prevent execution of the STC.W instruction from failing in this case is to allocate only one mutual exclusion variable per 32 bytes of memory. If more than one mutual exclusion variable is allocated in a 32-byte range, thrashing might occur when an attempt is made to acquire a lock on a mutual exclusion variable.



(2) Sample code

The sample code of a spinlock executed by using the LDL.W and STC.W instructions is shown below.

Lock acquisition

| | mov | lock_adr, | r20 |
|---------|--------|--------------|-----|
| Lock: | ldl.w | [r20], r21 | |
| | cmp | r0, r21 | |
| | bnz | Lock_wait | |
| | mov | 1, r21 | |
| | stc.w | r21, [r20] | |
| | cmp | r0, r21 | |
| | bnz | Lock_success | |
| Lock_wa | it: | | |
| | snooze | | |
| | br | Lock | |
| Lock_su | ccess: | | |

Lock release

st.w r0, 0[r20]



5.3.3 Performing Mutual Exclusion by Using the SET1 Instruction

The SET1 instruction can be used to perform mutual exclusion over multiple data arrays. By executing the SET1 instruction on the same bit in the memory and then checking the PSW.Z flag, which indicates the execution result, it can be determined whether lock acquisition succeeded or failed.

CAUTIONS

- Depending on the hardware specifications, the system performance might drop if exclusive control is executed frequently by using the SET1 instruction, because this causes the bus to be occupied for a long time. It is therefore recommended to execute exclusive control by using the LDL/STC instructions as much as possible.
- 2. When performing mutual exclusion by using the SET1 instruction, to prevent the problem of excessive bus occupancy described in Caution 1 above, execute the snooze instruction before attempting to acquire a lock again after lock acquisition has failed, and adjust the lock acquisition loop execution interval.

(1) Sample code

The sample code of a spinlock executed by using the SET1 instruction is shown below.

Lock acquisition

| | mov | lock_adr, r20 | | | | |
|----------|---------------|---------------|--|--|--|--|
| Lock: | set1 | 0, 0[r20] | | | | |
| | bz | Lock_success | | | | |
| | snooze | | | | | |
| | br | Lock | | | | |
| Lock_suc | Lock_success: | | | | | |

Lock release



5.3.4 Performing Mutual Exclusion by Using the CAXI Instruction

The CAXI instruction can be used to perform mutual exclusion over multiple data arrays. By executing the CAXI instruction on the same word in the memory and then checking the destination register, it can be determined whether lock acquisition succeeded or failed.

CAUTIONS

- Depending on the hardware specifications, the system performance might drop if exclusive control is executed frequently by using the CAXI instruction, because this causes the bus to be occupied for a long time. It is therefore recommended to execute exclusive control by using the LDL/STC instructions as much as possible.
- 2. When performing mutual exclusion by using the CAXI instruction, to prevent the problem of excessive bus occupancy described in Caution 1 above, execute the snooze instruction before attempting to acquire a lock again after lock acquisition has failed, and adjust the lock acquisition loop execution interval.

(1) Sample code

The sample code of a spinlock executed by using the CAXI instruction is shown below.

Lock acquisition

| | mov | lock_adr, r20 | | | |
|----------|---------------|----------------|--|--|--|
| Lock: | mov | 1, r21 | | | |
| | caxi | [r20], r0, r21 | | | |
| | bz | Lock_success | | | |
| | snooze | | | | |
| | br | Lock | | | |
| Lock_suc | lock_success: | | | | |

Lock release

| SL.W | ru, u[r2u] |
|------|------------|
| | |



Section 6 COPROCESSOR

6.1 Floating-Point Operation

The floating-point unit (FPU) operates as the CPU coprocessor, and executes floating-point instructions.

Either single-precision (32-bit) or double-precision (64-bit) data can be used. In addition, the conversion between a floating point type and an integer type is possible.

The FPU of this CPU conforms to ANSI/IEEE standard 754-2008 (IEEE Standard for Floating-Point Arithmetic).

6.1.1 Configuration of Floating-Point Operation Function

(1) Not implemented

If the floating-point operation function is not implemented, all the floating-point instructions cannot be used. If an attempt is made to execute such an instruction, a coprocessor unusable exception occurs. In addition, the operation of all the floating-point system registers is undefined. Therefore, do not manipulate these registers by LDSR and STSR.

(2) Implementing only single precision

If only the floating-point operation function with single precision is implemented, only floating-point instructions classified as single precision^{*1} can be used. If an attempt is made to execute a floating-point instruction classified as double precision^{*2}, a coprocessor unusable exception occurs. All the floating-point system registers supply the function described in **Section 3.4, FPU Function Registers**.

- **Note 1.** The single-precision floating-point instruction is the instruction described as (Single) in the description of each instruction in **Section 7.4.4, Floating-Point Instruction Set**.
- **Note 2.** The double-precision floating-point instruction is the instruction described as (Double) in the description of each instruction in **Section 7.4.4, Floating-Point Instruction Set**.

(3) Implementing single precision and double precision

All the floating-point instructions can be used when floating-point instructions of single precision and double precision are implemented. All the floating-point system registers supply the functions described in **Section 3.4, FPU Function Registers**.



6.1.2 Data Types

(1) Floating-point format

The FPU supports 32-bit (single precision) and 64-bit (double precision) IEEE754 floating-point operations.

The single-precision floating-point format consists of a 24-bit signed fraction (s + f) and an 8-bit exponent (e), as shown in **Figure 6.1**.

| 31 30 | 0 23 | 22 22 | 0 |
|-------|----------|----------|---|
| S | е | f | |
| Sign | Exponent | Fraction | |
| 1 | 8 | 23 | |

Figure 6.1 Single-precision Floating-point Format

The double-precision floating-point format consists of a 53-bit signed fraction (s + f) and an 11-bit exponent (e), as shown in **Figure 6.2**.



Figure 6.2 Double-precision Floating-Point Format

A numerical value in the floating-point format includes the following three areas.

- Sign bit: s
- Exponent: e = E + bias value
- Fraction: $f = .b_1b_2...b_{P-1}$ (value lower than the first decimal place)

The bias value for the single-precision format is 127. For double-precision format, the bias value is 1023.

The range of the exponent value E when unbiased covers all integers from Emin to Emax, along with two reserved values, Emin -1 (± 0 or subnormal number), and Emax +1 ($\pm \infty$ or NaN: not-a-number). A numeric value other than 0 is represented in one format, depending on the single-precision and double-precision formats.

The numeric value (v) represented in this format can be calculated by the expression shown in **Table 6.1**.



| | <u> </u> | | |
|----------------------|----------------------------------|------------------------------------|--|
| Туре | Calculation Expression | | |
| NaN (not-a-number) | If $E = Emax + 1$ and $f \neq 0$ | then v = NaN regardless of s | |
| ±∞ (infinite number) | If $E = Emax + 1$ and $f = 0$ | then $v = (-1)^{s_{\infty}}$ | |
| Normalized number | If $Emin \le E \le Emax$ | then $v = (-1)^{s} 2^{E} (1.f)$ | |
| Subnormal number | If $E = Emin - 1$ and $f \neq 0$ | then $v = (-1)^{s} 2^{Emin} (0.f)$ | |
| ±0 (zero) | If $E = Emin - 1$ and $f = 0$ | then $v = (-1)^{s}0$ | |

Table 6.1 Calculation Expression of Floating-Point Value

• NaN (not-a-number)

IEEE754 defines a floating-point value called NaN (not-a-number). Because this value is not a numerical value, it does not have any "greater than" or "less than" relationships to other values. If v is NaN in all of the floating-point formats, it might be either SignalingNaN (S-NaN) or QuietNaN (Q-NaN), depending on the value of the most significant bit of f. If the most significant bit of f is set, v is QuietNaN; if the most significant bit is cleared, it is SignalingNaN.

Table 6.2 shows the value of each parameter defined in floating-point formats.

Table 6.2 Floating-Point Formats and Parameter Values

| | Format | | |
|-------------------------------------|------------------|------------------|--|
| Parameter | Single Precision | Double Precision | |
| Emax | +127 | +1023 | |
| Emin | -126 | -1022 | |
| Bias value of exponent | +127 | +1023 | |
| Length of exponent (number of bits) | 8 | 11 | |
| Integer bits | Cannot be seen | Cannot be seen | |
| Length of fraction (number of bits) | 23 | 52 | |
| Length of format (number of bits) | 32 | 64 | |

Table 6.3 shows the minimum and maximum values that can be represented in floating-point formats.

| Table 6.3 F | loating-Point | Minimum an | d Maximum | Values |
|-------------|----------------------|------------|-----------|--------|
|-------------|----------------------|------------|-----------|--------|

| Туре | Value |
|-----------------------------------------------------------|---------------------------|
| Minimum value of single-precision floating point | 1.40129846e – 45 |
| Minimum value of single-precision floating point (normal) | 1.17549435e – 38 |
| Maximum value of single-precision floating point | 3.40282347e + 38 |
| Minimum value of double-precision floating point | 4.9406564584124654e - 324 |
| Minimum value of double-precision floating point (normal) | 2.2250738585072014e - 308 |
| Maximum value of double-precision floating point | 1.7976931348623157e + 308 |



(2) Fixed-point formats

The value of a fixed point is held in the format of 2's complement. **Figure 6.3** shows a 32-bit fixed-point format and **Figure 6.4** shows a 64-bit fixed-point format. No signed bits exist in the unsigned fixed-point format, and all bits represent the integer value.



Figure 6.3 32-bit Fixed-Point Format



Figure 6.4 64-bit Fixed-Point Format

(3) Expanded floating-point format

This CPU supports the 16-bit (half-precision) IEEE754 floating-point format as a floating-point format for storing data. The half-precision floating-point format is used to decrease the amount of data; it is not supported for arithmetic operations. Instructions are available for converting single-precision floating-point format data into half-precision floating-point data and vice-versa. The single-precision floating-point format consists of an 11-bit signed fraction (s + f) and a 5-bit exponent (e), as shown in **Figure 6.5**.







Like other floating-point formats, the numeric values represented in this format can be calculated by using the expressions shown in **Table 6.1**. The values of the parameters defined by the half-precision floating-point format are shown in **Table 6.4**.

| Parameter | Half Precision |
|-------------------------------------|----------------|
| Emax | +15 |
| Emin | -14 |
| Bias value of exponent | +15 |
| Length of exponent (number of bits) | 5 |
| Integer bits | Cannot be seen |
| Length of fraction (number of bits) | 10 |
| Length of format (number of bits) | 16 |

Table 6.5 shows the minimum and maximum values that can be represented in the half-precision floating-point format.

| Table 6.5 H | alf-Precision | Floating-Point | Minimum | and | Maximum | Values |
|-------------|---------------|-----------------------|---------|-----|---------|--------|
|-------------|---------------|-----------------------|---------|-----|---------|--------|

| Туре | Value |
|---------------------------------------------------------|------------------------|
| Minimum value of half-precision floating point | 5.96046e ⁻⁸ |
| Maximum value of half-precision floating point (normal) | 6.10352e ⁻⁵ |
| Maximum value of half-precision floating point | 65504 |

6.1.3 Register Set

For details about the register set, see Section 3.4, FPU Function Registers.

6.1.4 Floating-Point Instructions

Floating-point instructions are divided into single-precision instructions (single) and double-precision instructions (double).

For details about the floating-point instructions, see **Section 7.4**, **Floating-Point Instructions**.



6.1.5 Floating-Point Operation Exceptions

This section describes how the FPU processes floating-point operation exceptions.

(1) Types of exceptions

When floating-point operations or processing of operation results cannot be done using the ordinary method, a floating-point operation exception occurs.

One of the following two operations is performed when a floating-point operation exception has occurred.

• When exceptions are enabled

The cause bit is set in the floating-point configuration/status register (FPSR), and processing (by software) is passed to the exception handler routine.

• When exceptions are prohibited

The preservation bit is set in the floating-point configuration/status register (FPSR), an appropriate value (initial value) is stored in the FPU destination register, then execution is continued.

The FPU uses cause bits, enable bits, and preservation bits (status flags) to support the following five types of IEEE754-defined exception causes.

- Inexact operation (I)
- Overflow (O)
- Underflow (U)
- Division by zero (Z)
- Invalid operation (V)

A sixth type of exception cause is unimplemented operation (E), which causes an exception when a floating-point operation cannot be executed. This exception requires processing by software. An unimplemented operation exception (E) occurs when exceptions are always enabled, rather than by using properties, enable bits, or preservation bits.





Figure 6.6 shows the FPSR register bits that are used to support exceptions.

Figure 6.6 Cause, Enable, and Preservation Bits of FPSR Register

The five exceptions (V, Z, O, U, and I) defined by IEEE754 are enabled when the corresponding enable bits are set. When an exception occurs, if the corresponding enable bit has been set, the FPU sets the corresponding cause bit. If the exception can be acknowledged, processing is passed to the exception handler routine. If exceptions are prohibited, the exception corresponding preservation bit is set, and processing is not passed to the exception handler routine.



(2) Exception handling

When a floating-point operation exception occurs, the cause bits of the FPSR register indicate the cause of the floating-point operation exception.

(a) Status flag

A corresponding preservation bit is available for each IEEE754-defined exception. The preservation bit is set when the corresponding exception is prohibited but the exception condition has been detected. The preservation bit is set or reset whenever new values are written to the FPSR register by the LDSR instruction.

If an exception is prohibited by an enable bit, predetermined processing is performed by the FPU. This processing provides an initial value as the result, rather than a floating-point operation result. This initial value is determined according to the type of exception. For an overflow exception or underflow exception, the initial value also differs depending on the current rounding mode. **Table 6.6** shows the initial values provided for each of the FPU IEEE754-defined exceptions.

| Area | Description | Rounding Mode | Initial Value |
|------|-------------------------|---------------|------------------------------------------------------------------------------------|
| V | Invalid operation | _ | Quiet not-a-number (Q-NaN) |
| Z | Division by zero | _ | Correctly signed ∞ |
| 0 | Overflow | RN | ∞ with sign of intermediate result |
| | | RZ | Maximum normalized number with sign of intermediate result |
| | | RP | Negative overflow: Maximum negative normalized number Positive overflow: +∞ |
| | | RM | Positive overflow: Maximum positive normalized number Negative overflow: $-\infty$ |
| U | Underflow ^{*1} | RN*2 | 0 with sign of intermediate result |
| | | RZ | 0 with sign of intermediate result |
| | | RP | Positive underflow: Minimum positive normalized number Negative underflow: 0 |
| | | RM | Negative underflow: Minimum negative normalized number Positive underflow: 0 |
| | Inexact operation | | Rounded result |
| | | | |

 Table 6.6
 FPU Initial Values for IEEE754-Defined Exceptions

Note 1. If the FPSR.FS bit is cleared, an unimplemented operation exception (E) will occur if an underflow occurs in the rounded result; an underflow exception (U) will not occur. If the FS bit of the FPSR register is set, the flushed result is used as the default value

Note 2. If the rounding mode is RN and the FN bit of the FPSR register is set, flushing will occur in the direction of higher accuracy. For details, see Section 6.1.9, Flush to Nearest.



6.1.6 Exception Details

The following describes the conditions under which each of the FPU exceptions occurs and the FPU responses.

(1) Inexact exception (I)

In the following cases, the FPU detects an inexact exception.

- When the precision of the rounded result is dropped
- When the rounded result overflows while overflow exceptions are prohibited
- When the rounded result underflows while underflow exceptions are prohibited
- When the operand that is a subnormal number is flushed, neither an invalid operation exception (V) nor a division by zero exception (Z) is detected, and the other operands are not Q-NaN

CAUTION

If the FS bit of the FPSR register is cleared and the operation result underflows, an unimplemented operation exception (E) occurs. In such cases, the underflow exception is not detected, so the inexact exception is not detected either.

(a) If exception is enabled

The contents of the destination register are not changed, contents of the source register are saved, and an inexact exception occurs.

(b) If exception is not enabled

If no other exception occurs, the rounded result or the result that underflows or overflows is stored in the destination register.



(2) Invalid operation exception (V)

An invalid operation exception occurs when one of both of the operands is invalid.

- Arithmetic operation with S-NaN included in operands. The conditional move instruction (CMOV), absolute value (ABS), and arithmetic negation (NEG) are not handled as arithmetic operations, but minimum value (MIN) and maximum value (MAX) are handled as arithmetic operations.
- Multiplication: $\pm 0 \times \pm \infty$ or $\pm \infty \times \pm 0$
- Fused-multiply-add: $(\pm 0 \times \pm \infty) + c$ or $(\pm \infty \times \pm 0) + c$. But only if c is not Q-NaN.
- Addition/subtraction or multiply-add operation*¹:
 Addition of infinite values with different signs or subtraction of infinite values with the same sign
- Division: $\pm 0 \div \pm 0$ or $\pm \infty \div \pm \infty$
- Square root: When operand is less than 0
- Conversion to integer when source is outside of integer range.
- Comparison:With condition codes 8 to 15, if the operand is unordered (see **Table 7.10**, **Definitions of Condition Code Bits and Their Logical Inversions**)

Note 1. When the multiplication result is infinite or when adding or subtracting between infinities

(a) If exception is enabled

The contents of the destination register are not changed, contents of the source register are saved, and an invalid operation exception occurs.

(b) If exception is not enabled

If no other exception occurs, and the destination is a floating-point format, Q-NaN is stored in the destination register. If the destination has an integer format, see the operation result description of each instruction for the value to be stored in the destination register.

(3) Division by zero exception (Z)

A division by zero exception occurs when a divisor is 0 and a dividend is a finite number other than 0.

(a) If exception is enabled

The contents of the destination register are not changed, contents of the source register are saved, and a division by zero exception occurs.

(b) If exception is not enabled

If no other exception occurs, a correctly signed infinite number $(\pm \infty)$ is stored in the destination register.



(4) Overflow exception (O)

An overflow exception is detected if the exponent range is infinite and if the result of the rounded floating point is greater than maximum finite number in the destination format.

(a) If exception is enabled

The contents of the destination register are not changed, the contents of the source register are saved, and an overflow exception occurs.

(b) If exception is not enabled

If no other exception occurs, the initial value that is determined by the rounding mode and the sign of the intermediate result is stored in the destination register (see **Table 6.6, FPU Initial Values** for IEEE754-Defined Exceptions).

(5) Underflow exception (U)

If the operation result is -2^{Emin} to $+2^{\text{Emin}}$ (but not zero), an underflow exception is detected.

Although IEEE754 defines several methods for detecting an underflow, the same method should be used to detect underflows, regardless of the processing to be performed.

The following two methods can be used to detect an underflow for binary floating point numbers.

- The result calculated after rounding and using an infinite exponent range is not zero and is within $\pm 2^{\text{Emin}}$.
- The result calculated before rounding and using an infinite exponent range and precision is not zero and is within $\pm 2^{\text{Emin}}$.

In this CPU, an underflow is detected before rounding.

Or the rounded result is one of the following, an inexact result is detected.

• When a given result differs from the result calculated when the exponent range and precision are infinite)

In this CPU, the behavior when an inexact result is detected differs as follows depending on whether underflow exceptions are enabled or disabled:

(a) If exception is enabled

When the FS bit of the FPSR register has been set, if exceptions are enabled, an underflow exception (U) occurs. When the FS bit of the FPSR register has been set, if exceptions are not enabled but inexact exceptions are enabled, an inexact exception (I) occurs.

(b) If exception is not enabled

If the FS bit of the FPSR register has been set, the initial value determined according to the rounding mode and intermediate result value is stored in the destination register (see **Table 6.6**, **FPU Initial Values for IEEE754-Defined Exceptions**).

CAUTION

If the FS bit of the FPSR register has not been set, an unimplemented operation exception (E) occurs regardless of whether or not exceptions are enabled. Because an unimplemented operation exception (E) must occur, an underflow exception (U) does not occur.

(6) Unimplemented operation exception (E)

The E bit is set and an unimplemented operation exception (E) occurs when an abnormal operand or abnormal result that cannot be correctly processed by hardware has been detected. The operand and destination register contents do not change.

If the FS bit of the FPSR register has been set, an unimplemented operation exception (E) will not occur.

If the FS bit of the FPSR register has been cleared, an unimplemented operation exception (E) will occur under the following conditions (except for CMOVF.D, CMOVF.S, CMPF.D, CMPF.S, ABSF.D, ABSF.S, MAXF.D, MAXF.S, MINF.D, MINF.S, NEGF.D, and NEGF.S instructions).

- When the operand is a subnormal number
- When the operation result is a subnormal number, or an underflow has occurred

CAUTION

If the FS bit of the FPSR register is set to 1, an unimplemented operation exception (E) will not occur under any circumstances.



6.1.7 Saving and Returning Status

When a floating-point operation exception occurs, the PC and PSW are saved to the EIPC and EIPSW registers respectively, and the exception code is saved to the EIIC register.

When an EI level exception is acknowledged while processing a floating-point operation exception, an EIPC register override occurs, which prevents the returning to the instruction that caused the floating-point operation exception to occur. When acknowledgment of EI level exceptions is required, the contents of the EIPC, EIPSW, and EIIC registers must be saved, such as to a stack.

When a floating-point instruction is used in a floating-point operation exception handler routine, the FPSR and FPEPC registers will override if another floating-point operation exception occurred. In such cases, the FPSR and FPEPC registers should be saved at the start of the floating-point operation exception handler processing, and should be returned at the end of the handler processing.

The cause bits of the FPSR register hold the results from only one enabled exception. In any case, the previous results are held until the next enabled exception occurs.



6.1.8 Flushing Subnormal Numbers

This CPU can process subnormal numbers—very small numbers that are lower than the minimum normalized number—in one of the following two ways:

- Normalize the operand or operation result and continue executing arithmetic processing
- Generate an unimplemented operation exception (E) and execute exception handling

Executing software-based exception handling will obtain a more accurate result, but the amount of time required to obtain the result will vary depending on the input value. In control systems that require a real-time performance, therefore, this is usually unacceptable. In this case, it is important to obtain the result within a certain amount time rather than focus on accuracy.

(1) Normalize the subnormal numbers and continue executing arithmetic processing

By setting the FS bit of the FPSR register to 1, this CPU can normalize the operand or operation result to a specific value and continue executing arithmetic processing if a subnormal number is input as the operand or obtained as the operation result. At this time, extremely small differences in values might not appear in the operation result.

For the operand and operation result, the values to which subnormal numbers are flushed when the FS bit is set (1) are shown in **Table 6.7** and **Table 6.8** below.

| Table 6.7 | Rounding | Mode and | Flush | Value of | Input | Operand |
|-----------|----------|----------|-------|----------|-------|---------|
|-----------|----------|----------|-------|----------|-------|---------|

| | Rounding Mode and Value to Which Input Operand Is Flushed | | | | |
|---------------------------|--------------------------------------------------------------|----|----|----|--|
| Sign of Subnormal Operand | RN | RZ | RP | RM | |
| + | | | +0 | | |
| _ | -0 | | | | |

Table 6.8 Rounding Mode and Flush Value of Operation Result

| | Rounding Mode and Value to Which Operation Result Is Flushed | | | | |
|------------------------------------|-----------------------------------------------------------------|----|--------------------|--------------------|--|
| Sign of Subnormal Operation Result | RN ^{Note} | RZ | RP | RM | |
| + | +0 | +0 | +2 ^{Emin} | +0 | |
| - | -0 | -0 | -0 | -2 ^{Emin} | |

Note 1. If the rounding mode is RN and the FN bit of the FPSR register is set, flushing will occur in the direction of higher accuracy. For details, see **Section 6.1.9, Flush to Nearest**.

Whether an input operand that is a subnormal number has been flushed or not can be checked by referencing the IF bit of the FPSR register. Whether an operation result that is a subnormal number has been flushed or not can be checked by referencing the U bit of the FPSR register.

CAUTIONS

- 1. In control systems that require a real-time performance, it is recommended to always set the FS bit to 1.
- If the FS bit of the FPSR register is set (1), an unimplemented operation exception (E) will not occur under any circumstances.
- 3. Whether the operation result is a subnormal number is judged by using the value before rounding.
- 4. The IF bit of the FPSR register also accumulates and indicates information about flushing instructions that have caused a floating-point operation exception.

(2) Generate an unimplemented operation exception (E) and execute exception handling

By clearing the FS bit of the FPSR register to 0, an unimplemented operation exception (E) will occur if a subnormal number is input as the operand or obtained as the operation result. When an unimplemented operation exception occurs, software-based progressive underflow processing is performed in the floating-point operation exception handling routine, enabling a more accurate result to be obtained. In this case, however, a real-time processing performance might not be realized due to the software processing load.

(3) Instructions that can handle subnormal numbers

The following instructions can be executed without causing an unimplemented operation exception even if an operand that is a subnormal number is input while the FS bit of the FPSR register is 0.

- Conditional move instruction (CMOV), absolute value (ABS), arithmetic negation (NEG)
- Minimum value (MIN), maximum value (MAX), compare (CMPF)
- Conversion from half-precision to single-precision (CVTF.HS)

(4) Instructions that are not affected by flushing subnormal numbers

For the following instructions, flushing does not occur even an operand that is a subnormal number is input while the FS bit of the FPSR register is 1.

- Conditional move instruction (CMOV), absolute value (ABS), arithmetic negation (NEG)
- Minimum value (MIN), maximum value (MAX), compare (CMPF)
- Conversion from half-precision to single-precision (CVTF.HS)



6.1.9 Flush to Nearest

This CPU provides flush-to-nearest mode, a feature for flushing to the nearest number with higher accuracy when a flushing operation results subnormal number. Flush-to-nearest mode is enabled when the rounding mode is RN and the FN bit of the FPSR register is set (1). When this mode is used, the FPU determines the value to which to flush the subnormal number based on the number of the operation result and not just the sign. This feature has no effect in rounding modes other than RN or on the result of flushing an input operand.

| Table 6.9 | Rounding Mode | and Value to | Which Operation | Result is Flushed |
|-----------|----------------------|--------------|------------------|--------------------------|
| | ittouriding mou | | minori oporation | ittoodit io i idollod |

| | Rounding Mode and Value to Which Operation Result Is Flushed | | | | |
|---------------------------------------------------------------------|--------------------------------------------------------------|--------|----|--------------------|--------------------|
| Value of Subnormal Operation | RN | | | | |
| Result | FN = 1 | FN = 0 | RZ | RP | RM |
| $+2^{\text{Emin-1}} \le \text{Operation result} < +2^{\text{Emin}}$ | +2 ^{Emin} | +0 | +0 | +2 ^{Emin} | +0 |
| +0 \leq Operation result $<$ +2 ^{Emin-1} | +0 | | | | |
| $-2^{\text{Emin-1}} < \text{Operation result} \le -0$ | -0 | -0 | -0 | -0 | -2 ^{Emin} |
| $-2^{\text{Emin}} < \text{Operation result} \le -2^{\text{Emin-1}}$ | -2 ^{Emin} | | | | |
| CAUTION | | | | | |

Whether the operation result is a subnormal number is judged by using the value before rounding.



Section 7 INSTRUCTION

7.1 Opcodes and Instruction Formats

This CPU has two types of instructions: CPU instructions, which are defined as basic instructions, and coprocessor instructions, which are defined according to the application.

7.1.1 CPU Instructions

Instructions classified as CPU instructions are allocated in the opcode area other than the area used in the format of the coprocessor instructions shown in **Section 7.1.2, Coprocessor Instructions**.

CPU instructions are basically expressed in 16-bit and 32-bit formats. There are also several instructions that use option data to add bits, enabling the configuration of 48-bit and 64-bit instructions. For details, see the opcode of the relevant instruction in **Section 7.2.2, Basic Instruction Set**.

Opcodes in the CPU instruction opcode area that do not define significant CPU instructions are reserved for future function expansion and cannot be used. For details, see **Section 7.1.3, Reserved Instructions**.

(1) reg-reg instruction (Format I)

A 16-bit instruction format consists of a 6-bit opcode field and two general-purpose register specification fields.



(2) imm-reg instruction (Format II)

A 16-bit instruction format consists of a 6-bit opcode field, 5-bit immediate field, and a general-purpose register specification field.





(3) Conditional branch instruction (Format III)

A 16-bit instruction format consists of a 4-bit opcode field, 4-bit condition code field, and an 8-bit displacement field.



(4) 16-bit load/store instruction (Format IV)

A 16-bit instruction format consists of a 4-bit opcode field, a general-purpose register specification field, and a 7-bit displacement field (or 6-bit displacement field + 1-bit sub-opcode field).



In addition, a 16-bit instruction format consists of a 7-bit opcode field, a general-purpose register specification field, and a 4-bit displacement field.



(5) Jump instruction (Format V)

A 32-bit instruction format consists of a 5-bit opcode field, a general-purpose register specification field, and a 22-bit displacement field.





(6) 3-operand instruction (Format VI)

A 32-bit instruction format consists of a 6-bit opcode field, two general-purpose register specification fields, and a 16-bit immediate field.



(7) 32-bit load/store instruction (Format VII)

A 32-bit instruction format consists of a 6-bit opcode field, two general-purpose register specification fields, and a 16-bit displacement field (or 15-bit displacement field + 1-bit sub-opcode field).



(8) Bit manipulation instruction (Format VIII)

A 32-bit instruction format consists of a 6-bit opcode field, 2-bit sub-opcode field, 3-bit bit specification field, a general-purpose register specification field, and a 16-bit displacement field.



(9) Extended instruction format 1 (Format IX)

This is a 32-bit instruction format that has a 6-bit opcode field and two general-purpose register specification fields, and handles the other bits as a sub-opcode field.

CAUTION

Extended instruction format 1 might use part of the general-purpose register specification field or the sub-opcode field as a system register number field, condition code field, immediate field, or displacement field. For details, see the description of each instruction in Section 7.2.2, Basic Instruction Set.





(10) Extended instruction format 2 (Format X)

This is a 32-bit instruction format that has a 6-bit opcode field and uses the other bits as a subopcode field.

CAUTION

Extended instruction format 2 might use part of the general-purpose register specification field or the sub-opcode field as a system register number field, condition code field, immediate field, or displacement field. For details, see the description of each instruction in Section 7.2.2, Basic Instruction Set.



(11) Extended instruction format 3 (Format XI)

This is a 32-bit instruction format that has a 6-bit opcode field and three general-purpose register specification fields, and uses the other bits as a sub-opcode field.

CAUTION

Extended instruction format 3 might use part of the general-purpose register specification field or the sub-opcode field as a system register number field, condition code field, immediate field, or displacement field. For details, see the description of each instruction in Section 7.2.2, Basic Instruction Set.



(12) Extended instruction format 4 (Format XII)

This is a 32-bit instruction format that has a 6-bit opcode field and two general-purpose register specification fields, and uses the other bits as a sub-opcode field.

CAUTION

Extended instruction format 4 might use part of the general-purpose register specification field or the sub-opcode field as a system register number field, condition code field, immediate field, or displacement field. For details, see the description of each instruction in Section 7.2.2, Basic Instruction Set.





(13) Stack manipulation instruction format (Format XIII)

A 32-bit instruction format consists of a 5-bit opcode field, 5-bit immediate field, 12-bit register list field, 5-bit sub-opcode field, and one general-purpose register specification field (or 5-bit sub-opcode field).

The general-purpose register specification field is used as a sub-opcode filed, depending on the format of the instruction.

| 15 11 | 10 6 | 5 1 | 0 21 21 | 20 16 |
|------------|--------|-----|---------|-------|
| sub-opcode | opcode | imm | | reg2 |

(14) Load/store instruction 48-bit format (Format XIV)

This is a 48-bit instruction format that has a 6-bit opcode field, two general-purpose register specification fields, and a 23-bit displacement field, and uses the other bits as a sub-opcode field.





7.1.2 Coprocessor Instructions

Instructions in the following format are defined as coprocessor instructions.



Coprocessor instructions define the functions of each coprocessor.

(1) Coprocessor unusable exception

If an attempt is made to execute a coprocessor instruction defined by an opcode that refers to a nonexistent coprocessor or a coprocessor that cannot be used due to the operational status of the device, a coprocessor unusable exception (UCPOP) immediately occurs.

For details, see Section 2.4.3, Coprocessor Unusable Exceptions.

7.1.3 Reserved Instructions

An opcode reserved for future function extension and for which no instruction is defined is defined as a reserved instruction. It is defined by the hardware specifications that either of the following two types of operations is performed on the opcode of a reserved instruction.

- A reserved instruction exception occurs
- The reserved instruction is executed as an instruction

In this CPU, the following opcodes define the RIE instruction, which always causes a reserved instruction exception to occur.



7.2 Basic Instructions

7.2.1 Overview of Basic Instructions

(1) Load instructions

Execute data transfer from memory to register. The following instructions (mnemonics) are provided.

- (a) LD instructions
- LD.B : Load byte
- LD.BU : Load byte unsigned
- LD.DW : Load double word
- LD.H : Load halfword
- LD.HU : Load halfword unsigned
- LD.W : Load word
- (b) SLD instructions
- SLD.B : Short format load byte
- SLD.BU : Short format load byte unsigned
- SLD.H : Short format load halfword
- SLD.HU : Short format load halfword unsigned
- SLD.W : Short format load word

(2) Store instructions

Execute data transfer from register to memory. The following instructions (mnemonics) are provided.

- (a) ST instructions
- ST.B : Store byte
- ST.DW : Store double word
- ST.H : Store halfword
- ST.W : Store word
- (b) SST instructions
- SST.B : Short format store byte
- SST.H : Short format store halfword
- SST.W : Short format store word

(3) Multiply instructions

Execute multiplication in one clock cycle with the on-chip hardware multiplier. The following instructions (mnemonics) are provided.

- MUL : Multiply word
- MULH : Multiply halfword
- MULHI : Multiply halfword immediate
- MULU : Multiply word unsigned


(4) Multiply-accumulate instructions

After a multiplication operation, a value is added to the result. The following instructions (mnemonics) are available.

- MAC : Multiply and add word
- MACU : Multiply and add word unsigned

(5) Arithmetic instructions

Add, subtract, transfer, or compare data between registers. The following instructions (mnemonics) are provided.

- ADD : Add
- ADDI : Add immediate
- CMP : Compare
- MOV : Move
- MOVEA : Move effective address
- MOVHI : Move high halfword
- SUB : Subtract
- SUBR : Subtract reverse

(6) Conditional arithmetic instructions

Add and subtract operations are performed under specified conditions. The following instructions (mnemonics) are available.

- ADF : Add on condition flag
- SBF : Subtract on condition flag

(7) Saturated operation instructions

Execute saturated addition and subtraction. If the operation result exceeds the maximum positive value (7FFF FFFF_{H}), 7FFF FFFF_{H} returns. If the operation result exceeds the maximum negative value (8000 0000_H), 8000 0000_H returns. The following instructions (mnemonics) are provided.

- SATADD : Saturated add
- SATSUB : Saturated subtract
- SATSUBI : Saturated subtract immediate
- SATSUBR : Saturated subtract reverse

(8) Logical instructions

Include logical operation instructions. The following instructions (mnemonics) are provided.

- AND : AND
- ANDI : AND immediate
- NOT : NOT
- OR : OR

- ORI : OR immediate
- TST : Test
- XOR : Exclusive OR
- XORI : Exclusive OR immediate

(9) Data manipulation instructions

Include data manipulation instructions and shift instructions with arithmetic shift and logical shift. Operands can be shifted by multiple bits in one clock cycle through the on-chip barrel shifter. The following instructions (mnemonics) are provided.

- BINS : Bitfield insert
- BSH : Byte swap halfword
- BSW : Byte swap word
- CMOV : Conditional move
- HSH : Halfword swap halfword
- HSW : Halfword swap word
- ROTL : Rotate left
- SAR : Shift arithmetic right
- SASF : Shift and set flag condition
- SETF : Set flag condition
- SHL : Shift logical left
- SHR : Shift logical right
- SXB : Sign-extend byte
- SXH : Sign-extend halfword
- ZXB : Zero-extend byte
- ZXH : Zero-extend halfword

(10) Bit search instructions

The specified bit values are searched among data stored in registers.

- SCH0L : Search zero from left
- SCH0R : Search zero from right
- SCH1L : Search one from left
- SCH1R : Search one from right

(11) Divide instructions

Execute division operations. Regardless of values stored in a register, the operation can be performed using a constant number of steps. The following instructions (mnemonics) are provided.

- DIV : Divide word
- DIVH : Divide halfword



- DIVHU : Divide halfword unsigned
- DIVU : Divide word unsigned

(12) High-speed divide instructions

These instructions perform division operations. The number of valid digits in the quotient is determined in advanced from values stored in a register, so the operation can be performed using a minimum number of steps. The following instructions (mnemonics) are provided.

- DIVQ : Divide word quickly
- DIVQU : Divide word unsigned quickly

(13) Branch instructions

Include unconditional branch instructions (JARL, JMP, and JR) and a conditional branch instruction (Bcond) which accommodates the flag status to switch controls. Program control can be transferred to the address specified by a branch instruction. The following instructions (mnemonics) are provided.

- Bcond (BC, BE, BGE, BGT, BH, BL, BLE, BLT, BN, BNC, BNE, BNH, BNL, BNV, BNZ, BP, BR, BSA, BV, BZ) : Branch on condition code
- JARL : Jump and register link
- JMP : Jump register
- JR : Jump relative

(14) Loop instruction

• LOOP : Loop

(15) Bit manipulation instructions

Execute logical operation on memory bit data. Only a specified bit is affected. The following instructions (mnemonics) are provided.

- CLR1 : Clear bit
- NOT1 : Not bit
- SET1 : Set bit
- TST1 : Test bit

(16) Special instructions

Include instructions not provided in the categories of instructions described above. The following instructions (mnemonics) are provided.

- CALLT : Call with table look up
- CAXI : Compare and exchange for interlock
- CLL : Clear load link
- CTRET : Return from CALLT
- DI : Disable interrupt
- DISPOSE : Function dispose
- EI : Enable interrupt



- EIRET : Return from trap or interrupt
- FERET : Return from trap or interrupt
- FETRAP : Software trap
- HALT : Halt
- LDSR : Load system register
- LDL.W : Load linked word
- NOP : No operation
- POPSP : Pop registers from stack
- PREPARE : Function prepare
- PUSHSP : Push registers from stack
- RIE : Reserved instruction exception
- SNOOZE : Snooze
- STSR : Store system register
- STC.W : Store conditional word
- SWITCH : Jump with table look up
- SYNCE : Synchronize exceptions
- SYNCI : Synchronize memory for instruction writers
- SYNCM : Synchronize memory
- SYNCP : Synchronize pipeline
- SYSCALL : System call
- TRAP : Trap



7.2.2 Basic Instruction Set

This section details each instruction, dividing each mnemonic (in alphabetical order) into the following items.

- Instruction format : Indicates how the instruction is written and its operand(s) (for symbols, see **Table 7.1**).
- Operation : Indicates the function of the instruction (for symbols, see Table 7.2).
- Format : Indicates the instruction format (see Section 7.1, Opcodes and Instruction Formats).
- Opcode : Indicates the bit field of the instruction opcode (for symbols, see **Table 7.3**).
- Flag : Indicates the change of flags of PSW (program status word) after the instruction execution. "0" is to clear (reset), "1" to set, and "—" to remain unchanged.
- Description : Describes the operation of the instruction.
- Supplement : Provides supplementary information on the instruction.
- Caution : Provides precautionary notes.

Table 7.1 Conventions of Instruction Format

| Symbol | Meaning | |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| reg1 | General-purpose register (as source register) | |
| reg2 | General-purpose register (primarily as destination register with some as source registers) | |
| reg3 | General-purpose register (primarily used to store the remainder of a division result and/or the higher 32 bits of a multiplication result) | |
| bit#3 | 3-bit data to specify bit number | |
| imm × | x-bit immediate data | |
| disp × | x-bit displacement data | |
| regID | System register number | |
| sellD | System register group number | |
| vector × | Data to specify vector (x indicates the bit size) | |
| cond | Condition code (see Table 7.4 Condition Codes) | |
| CCCC | 4-bit data to specify condition code (see Table 7.4 Condition Codes) | |
| sp | Stack pointer (r3) | |
| ер | Element pointer (r30) | |
| list12 | Lists of registers | |
| rh-rt | Indicates multiple general-purpose registers, from the general-purpose register indicated by <i>rh</i> to the general-purpose register indicated by <i>rt</i> . | |



| Symbol | Meaning |
|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| ← | Assignment |
| GR [a] | Value stored in general-purpose register a |
| SR [a, b] | Value stored in system register (RegID = a, SeIID = b) |
| (n:m) | Bit selection. Select from bit <i>n</i> to bit <i>m</i> . |
| zero-extend (n) | Zero-extends "n" to word |
| sign-extend (n) | Sign-extends "n" to word |
| load-memory (a, b) | Reads data of size b from address a |
| store-memory (a, b, c) | Writes data b of size c to address a |
| extract-bit (a, b) | Extracts value of bit b of data a |
| set-bit (a, b) | Sets value of bit b of data a |
| not-bit (a, b) | Inverts value of bit b of data a |
| clear-bit (a, b) | Clears value of bit b of data a |
| saturated (n) | Performs saturated processing of "n." If $n \ge 7FFF FFFF_H$, $n = 7FFF FFFF_H$. If $n \le 8000\ 0000_H$, $n = 8000\ 0000_H$. |
| result | Outputs results on flag |
| Byte | Byte (8 bits) |
| Halfword | Halfword (16 bits) |
| Word | Word (32 bits) |
| == | Comparison (true upon a match) |
| != | Comparison (true upon a mismatch) |
| + | Add |
| - | Subtract |
| | Bit concatenation |
| × | Multiply |
| ÷ | Divide |
| % | Remainder of division results |
| AND | AND |
| OR | OR |
| XOR | Exclusive OR |
| NOT | Logical negate |
| logically shift left by | Logical left-shift |
| logically shift right by | Logical right-shift |
| arithmetically shift right by | Arithmetic right-shift |

Table 7.2 Conventions of Operation



| Symbol | Meaning |
|--------|-----------------------------------------------------------------------------|
| R | 1-bit data of code specifying reg1 or regID |
| r | 1-bit data of code specifying reg2 |
| w | 1-bit data of code specifying reg3 |
| D | 1-bit data of displacement (indicates higher bits of displacement) |
| d | 1-bit data of displacement |
| I | 1-bit data of immediate (indicates higher bits of immediate) |
| i | 1-bit data of immediate |
| V | 1-bit data of code specifying vector (indicates higher bits of vector) |
| v | 1-bit data of code specifying vector |
| CCCC | 4-bit data for condition code specification (See Table 7.4 Condition Codes) |
| bbb | 3-bit data for bit number specification |
| L | 1-bit data of code specifying general-purpose register in register list |
| S | 1-bit data of code specifying EIPC/FEPC, EIPSW/FEPSW in register list |
| Р | 1-bit data of code specifying PSW in register list |

Table 7.3 Conventions of Opcode

Table 7.4Condition Codes

| Condition Code (cccc) | Condition Name | Condition Formula |
|-----------------------|----------------|------------------------|
| 0000 | V | OV = 1 |
| 1000 | NV | OV = 0 |
| 0001 | C/L | CY = 1 |
| 1001 | NC/NL | CY = 0 |
| 0010 | Z | Z = 1 |
| 1010 | NZ | Z = 0 |
| 0011 | NH | (CY or Z) = 1 |
| 1011 | Н | (CY or Z) = 0 |
| 0100 | S/N | S = 1 |
| 1100 | NS/P | S = 0 |
| 0101 | Т | Always (Unconditional) |
| 1101 | SA | SAT = 1 |
| 0110 | LT | (S xor OV) = 1 |
| 1110 | GE | (S xor OV) = 0 |
| 0111 | LE | ((S xor OV) or Z) = 1 |
| 1111 | GT | ((S xor OV) or Z) = 0 |

<Arithmetic instruction>

| Α | D | D |
|---|---|---|
|---|---|---|

Add register/immediate

Add

| [Instruction format] | (1) ADD reg1, reg2 |
|----------------------|---------------------------------------------------------------------------------------------|
| | (2) ADD imm5, reg2 |
| | |
| [Operation] | (1) $GR [reg2] \leftarrow GR [reg2] + GR [reg1]$ |
| | (2) GR [reg2] \leftarrow GR [reg2] + sign-extend (imm5) |
| | |
| [Format] | (1) Format I |
| | (2) Format II |
| | |
| [Opcode] | |
| [-[] | 15 0 |
| | (1) rrrr001110RRRR |
| | |
| | $\begin{array}{c} 15 \\ (2) \end{array}$ |
| | |
| | |
| [Flags] | |
| | CY "1" if a carry occurs from MSB; otherwise, "0". |
| | OV "1" if overflow occurs; otherwise, "0". |
| | S "1" if the operation result is negative; otherwise, "0". |
| | Z "1" if the operation result is "0"; otherwise, "0". |
| | SAT — |
| | |
| [Description] | (1) Adds the word data of general-purpose register reg1 to the word data of general-purpose |
| | register reg2 and stores the result in general-purpose register reg2. General-purpose |
| | register reg1 is not affected. |
| | (2) Adds the 5-bit immediate data, sign-extended to word length, to the word data of |
| | general-purpose register reg2 and stores the result in general-purpose register reg2. |



<Arithmetic instruction>

| ADDI | | | Add immediate |
|----------------------|---------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| | | | Add immediate |
| [Instruction format] | ADDI im | um16, reg1, reg2 | |
| [Operation] | GR [reg2] | \leftarrow GR [reg1] + sign-extend (imm16) | |
| [Format] | Format V | Ι | |
| [Opcode] | 15 rrrrrl1 | 031 16 LOOOORRRRR iiiiiiiiiiiiiiiiii | |
| [Flags] | CY OV S Z SAT | "1" if a carry occurs from MSB; otherwise, "0". "1" if overflow occurs; otherwise, "0". "1" if the operation result is negative; otherwise, "0". "1" if the operation result is "0"; otherwise "0". | |
| [Description] | Adds the purpose re register re | 16-bit immediate data, sign-extended to word length, to the word dat egister reg1 and stores the result in general-purpose register reg2. Ge g1 is not affected. | a of general- neral-purpose |



<Conditional Operation Instructions>

| ADF | | | | | | Add on condition flag |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | | | | | Conditional add |
| [Instruction format] | ADF cccc, | reg1, reg2 | e, reg3 | | | |
| [Operation] | if conditions are satisfied then GR [reg3] \leftarrow GR [reg1] + GR [reg2] +1 else GR [reg3] \leftarrow GR [reg1] + GR [reg2] +0 | | | | | |
| [Format] | Format XI | | | | | |
| [Opcode] | 15 rrrrr111 | l111rrrf | 031 RR wwwww011101c | 16 ccc0 | | |
| [Flags] | CY OV S Z SAT | "1" if a ("1" if ov "1" if the "1" if the — | carry occurs from MSB; erflow occurs; otherwise e operation result is nega e operation result is "0"; | otherwise, "0". , "0". ative; otherwise, otherwise, "0". | "0". | |
| [Description] | Adds 1 to the of general-period of general-period of the condition Condition Code | he result of purpose reg condition ition specifi gister reg1 general-pu rpose regis on in the fo Name | f adding the word data gister reg2 and stores specified as condition fied as condition code is added to the word d irpose register reg3. sters reg1 and reg2 are llowing table as [cccc Condition Formula | a of general-pu the result of ac code "cccc" is "cccc" is not s ata of general- e not affected. c]. (cccc is not Condition Code | rpose regist Idition in ge s satisfied. satisfied, the purpose reg Designate o equal to 110 Name | ter reg1 to the word data eneral-purpose register e word data of general- ister reg2, and the result one of the condition 01.) Condition Formula |
| | 0000 | V | OV = 1 | 0100 | S/N | S = 1 |
| | 1000 | NV | OV = 0 | 1100 | NS/P | S = 0 |

0001

1001

0010

1010

0011

1011

C/L

Ζ

NZ

NH

Н

NC/NL



CY = 1

CY = 0

Z = 1

Z = 0

(CY or Z) = 1

(CY or Z) = 0

Т

LT

GE

LE

GT

Setting prohibited

Always (Unconditional)

(S xor OV) = 1

(S xor OV) = 0

((S xor OV) or Z) = 1

((S xor OV) or Z) = 0

0101

0110

1110

0111

1111

(1101)

<Logical instruction>

| AND | | | AND |
|----------------------|---------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| [Instruction format] | AND reg | 1, reg2 | |
| [Operation] | GR [reg2] | ← GR [reg2] AND GR [reg1] | |
| [Format] | Format I | | |
| [Opcode] | 15 rrrr00 | 0)1010RRRRR | |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". — | |
| [Description] | ANDs the register re reg1 is no | word data of general-purpose register reg2 with the word data of general-purp g1 and stores the result in general-purpose register reg2. General-purpose regist t affected. | ose ster |



<Logical instruction>

| ANDI | | | AND immediate |
|----------------------|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| | | | |
| [Instruction format] | ANDI in | nm16, reg1, reg2 | |
| [Operation] | GR [reg2 |] \leftarrow GR [reg1] AND zero-extend (imm16) | |
| [Format] | Format V | 1 | |
| [Opcode] | 15 rrrrrl | 031 16 10110RRRRR iiiiiiiiiiiiiiii | |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". — | |
| [Description] | ANDs the extended purpose r | e word data of general-purpose register reg1 with the 16-bit immedia to word length, and stores the result in general-purpose register reg2 egister reg1 is not affected. | ate data, zero- 2. General- |



<Branch instruction>

| Bcond | Branch on condition code with 9-bit displacemen |
|----------------------|-----------------------------------------------------------------------------------------------------|
| L | Conditional branch |
| [Instruction format] | (1) Bcond disp9 |
| | (2) Bcond disp17 |
| [Operation] | (1) if conditions are satisfied |
| | then $PC \leftarrow PC + sign-extend$ (disp9) |
| | (2) if conditions are satisfied |
| | then $PC \leftarrow PC + sign-extend (disp17)$ |
| [Format] | (1) Format III |
| | (2) Format VII |
| [Opcode] | 15 0 |
| | (1) ddddd1011dddcccc |
| | ddddddd is the higher 8 bits of disp9. |
| | cccc is the condition code of the condition indicated by cond (see Table 7.5, Bcond |
| | Instructions). |
| | <u>15</u> 0 <u>31</u> 16 |
| | (2) 00000111111DCCCC dddddddddddddd |
| | ddddddddddddd is the higher 16 bits of disp17. |
| | cccc is the condition code of the condition indicated by cond. (For details, see Table 7.5 , |
| | Bcond Instructions). |
| [Flags] | |
| | CY — |
| | S — |
| | Z — |
| | SAT — |



[Description]

(1) Checks each PSW flag specified by the instruction and branches if a condition is met; otherwise, executes the next instruction. The PC of branch destination is the sum of the current PC value and the 9-bit displacement (= 8-bit immediate data shifted by 1 and sign-extended to word length).

(2) Checks each PSW flag specified by the instruction and then adds the result of logically shifting the 16-bit immediate data 1 bit to the left and sign-extending it to word length to the current PC value if the conditions are satisfied. Control is then transferred. If the conditions are not satisfied, the system continues to the next instruction. BR (0101) cannot be specified as the condition code.

[Supplement]

Bit 0 of the 9-bit displacement is masked to "0". The current PC value used for calculation is the address of the first byte of this instruction. The displacement value being "0" signifies that the branch destination is the instruction itself.

| Instr | uction | Condition Code (cccc) | Flag Status | Branch Condition |
|----------|--------|--------------------------|------------------------|-------------------------------------------------------------------------------------|
| Signed | BGE | 1110 | (S xor OV) = 0 | Greater than or equal to signed |
| integer | BGT | 1111 | ((S xor OV) or Z) = 0 | Greater than signed |
| | BLE | 0111 | ((S xor OV) or Z) = 1 | Less than or equal to signed |
| | BLT | 0110 | (S xor OV) = 1 | Less than signed |
| Unsigned | BH | 1011 | (CY or Z) = 0 | Higher (Greater than) |
| integer | BL | 0001 | CY = 1 | Lower (Less than) |
| | BNH | 0011 | (CY or Z) = 1 | Not higher (Less than or equal) |
| | BNL | 1001 | CY = 0 | Not lower (Greater than or equal) |
| Common | BE | 0010 | Z = 1 | Equal |
| | BNE | 1010 | Z = 0 | Not equal |
| Others | BC | 0001 | CY = 1 | Carry |
| | BF | 1010 | Z = 0 | False |
| | BN | 0100 | S = 1 | Negative |
| | BNC | 1001 | CY = 0 | No carry |
| | BNV | 1000 | OV = 0 | No overflow |
| | BNZ | 1010 | Z = 0 | Not zero |
| | BP | 1100 | S = 0 | Positive |
| | BR | 0101 | _ | Always (unconditional) Cannot be specified when using instruction format (2). |
| | BSA | 1101 | SAT = 1 | Saturated |
| | BT | 0010 | Z = 1 | True |
| | BV | 0000 | OV = 1 | Overflow |
| | BZ | 0010 | Z = 1 | Zero |

Table 7.5Bcond Instructions

CAUTIONS

- The branch condition loses its meaning if a conditional branch instruction is executed on a signed integer (BGE, BGT, BLE, or BLT) when the saturated operation instruction sets "1" to the SAT flag. In normal operations, if an overflow occurs, the S flag is inverted (0 → 1 or 1 → 0). This is because the result is a negative value if it exceeds the maximum positive value and it is a positive value if it exceeds the maximum negative value. However, when a saturated operation instruction is executed, and if the result exceeds the maximum positive value, the result is saturated with a positive value; if the result exceeds the maximum negative value, the result is saturated with a negative value. Unlike the normal operation, the S flag is not inverted even if an overflow occurs.
- 2. For Bcond disp17 (instruction format (2)), BR (0101) cannot be specified as the condition code.



| BINS | Bitfield Inser |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Insert bit in registe |
| [Instruction format] | BINS reg1, pos, width, reg2 |
| [Operation] | $GR [reg2] \leftarrow GR [reg2](31:width+pos) \parallel GR [reg1](width-1:0) \parallel GR [reg2](pos-1:0)$ |
| [Format] | Format IX |
| [Opcode] | $\begin{array}{cccccccccccccccccccccccccccccccccccc$ |
| | 15 031 16 rrrr111111RRRRR MMMMK0001011LLL0 (msb ≥ 16, lsb < 16) |
| | 15 031 16 rrrr111111RRRRR MMMMK0001101LLL0 (msb < 16, lsb < 16) |
| | Most significant bit of field to be updated: msb = pos+width-1 Least significant bit of field to be updated: lsb = pos MMMM = lower 4 bits of msb, KLLL = lower 4 bits of lsb |
| [Flags] | CY — OV 0 S "1" if operation result word data MSB is "1": otherwise "0" |
| | Z "1" if operation result is "0"; otherwise, "0". SAT — |
| [Description] | Loads the lower width bits in general-purpose register reg1 and stores them from the bit position bit pos + width -1 in the specified field in general-purpose register reg2 in bit pos. This instruction does not affect any fields in general-purpose register reg2 except the specified field, nor does it affect general-purpose register reg1. |
| [Supplement] | The most significant bit (msb: bit pos + width -1) in the field in general-purpose register reg2 to be updated and the least significant bit (lsb: bit pos) in this field are specified by using, respectively the lower 4 bits, the MMMM and KLLL fields in the BINS instruction. The lower 3 bits of the sub-opcode field (bits 23 to 21) differ depending on the msb and lsb values. The operation is undefined if msb < lsb. |

RENESAS

| BSH | | Byte swap halfword |
|----------------------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Byte swap of halfword data |
| [Instruction format] | BSH reg | 2, reg3 |
| [Operation] | GR [reg3 |] \leftarrow GR [reg2] (23:16) GR [reg2] (31:24) GR [reg2] (7:0) GR [reg2] (15:8) |
| [Format] | Format X | П |
| [Opcode] | 15 rrrrl | 031 16 1111100000 wwww01101000010 |
| [Flags] | CY OV S Z SAT | "1" when there is at least one byte value of zero in the lower halfword of the operation result; otherwise; "0". "1" if operation result word data MSB is "1"; otherwise, "0". "1" when lower halfword of operation result is "0"; otherwise, "0". |
| [Description] | Executes | endian swap. |



| BSW | | Byte swap word |
|----------------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Byte swap of word data |
| [Instruction format] | BSW reg | g2, reg3 |
| [Operation] | GR [reg3 | GR [reg2] (7:0) GR [reg2] (15:8) GR [reg2] (23:16) GR [reg2] (31:24) |
| [Format] | Format X | |
| [Opcode] | 15 rrrrrl | 031 16 1111100000 wwww01101000000 |
| [Flags] | CY OV S Z SAT | "1" when there is at least one byte value of zero in the word data of the operation result; otherwise; "0". "1" if operation result word data MSB is "1"; otherwise, "0". "1" if operation result word data is "0"; otherwise, "0". |
| [Description] | Executes | endian swap. |



| CALLT | Call with table look up |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Subroutine call with table look up |
| [Instruction format] | CALLT imm6 |
| [Operation] | $CTPC \leftarrow PC + 2 \text{ (return PC)}$ $CTPSW(4:0) \leftarrow PSW(4:0)$ adr $\leftarrow CTBP + \text{zero-extend (imm6 logically shift left by 1)}^{*1}$ $PC \leftarrow CTBP + \text{zero-extend (Load-memory (adr, Half-word))}$ |
| | Caution 1. An MDP exception might occur depending on the result of address calculation. |
| [Format] | Format II |
| [Opcode] | 15 0 000001000iiiii |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | The following steps are taken. |
| | Transfers the contents of both return PC and PSW to CTPC and CTPSW. Adds the CTBP value to the 6-bit immediate data, logically left-shifted by 1, and zero-extended to word length, to generate a 32-bit table entry address. |
| | (3) Loads the halfword entry data of the address generated in step (2) and zero-extend to word length. |
| | (4) Adds the CTBP value to the data generated in step (3) to generate a 32-bit target address. |
| | (5) Jumps to the target address. |
| | |



CAUTIONS

- 1. When an exception occurs during CALLT instruction execution, the execution is aborted after the end of the read/write cycle.
- 2. Memory protection is performed when executing a memory read operation to read the CALLT instruction table. When memory protection is enabled, the data for generating a target address from a table allocated in an area to which access from a user program is prohibited cannot be loaded



Compare and exchange for interlock CAXI Comparison and swap [Instruction format] CAXI [reg1], reg2, reg3 adr \leftarrow GR[reg1]^{*1} [Operation] token ← Load-memory (adr, Word) $result \leftarrow GR[reg2] - token$ If result == 0then Store-memory (adr, GR[reg3], Word) $GR[reg3] \leftarrow token$ else Store-memory (adr, token, Word) $GR[reg3] \leftarrow token$ Caution 1. An MAE, or MDP exception might occur depending on the result of address calculation. [Format] Format XI [Opcode] 031 15 16 rrrr111111RRRRR wwww00011101110 [Flags] CY "1" if a borrow occurs in the result operation; otherwise, "0" ٥V "1" if overflow occurs in the result operation; otherwise, "0" S "1" if result is negative; otherwise, "0" Ζ "1" if result is 0; otherwise, "0" SAT [Description] Word data is read from the specified address and compared with the word data in generalpurpose register reg2, and the result is indicated by flags in the PSW. Comparison is performed by subtracting the read word data from the word data in general-purpose register reg2. If the comparison result is "0", word data in general-purpose register reg3 is stored in the generated address, otherwise the read word data is stored in the generated address. Afterward, the read word data is stored in general-purpose register reg3. General-purpose registers reg1 and reg2 are not affected.



CAUTIONS

- 1. This instruction provides an atomic guarantee aimed at exclusive control, and during the period between read and write operations, the target address is not affected by access due to any other cause.
- 2. The CAXI instruction is included for backward compatibility. If you are using a multi-core system and require an atomic guarantee, use the LDL.W and STC.W instructions.



| CLL | | Clear Load Link |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------|------------------------|
| | Clear atomic | manipulation link |
| [Instruction format] | CLL | |
| [Operation] | Llbit $\leftarrow 0$ | |
| [Format] | Format X | |
| [Opcode] | 15 031 16 1111111111111111111111111111111111 | |
| [Flags] | | |
| | CY — | |
| | S — | |
| | Z — | |
| | SAT — | |
| [Description] | The thread link generated by the LDL.W instruction is deleted. | |
| | For details about the link operation between the thread and core, see Section Performing Mutual Exclusion by Using the LDL.W and STC.W Inst | 1 5.3.2, tructions. |
| | CAUTION | |
| | In systems such as a multi-care system, how the CLL instruction operator den | anda on the |

In systems such as a multi-core system, how the CLL instruction operates depends on the system configuration of the product. For details, see the hardware manual of the product used.



| CLR1 | | Clear bit |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| | | BIT Clear |
| [Instruction format] | (1) CLR1 bit#3, disp16 [reg1] | |
| | (2) CLR1 reg2, [reg1] | |
| [Operation] | adr ← GR [reg1] + sign-extend (disp16)*1 token ← Load-memory (adr, Byte) Z flag ← Not (extract-bit (token, bit#3)) token ← clear-bit (token, bit#3) Store-memory (adr, token, Byte) | |
| | (2) adr ← GR [reg1]*1 token ← Load-memory (adr, Byte) Z flag ← Not (extract-bit (token, reg2)) token ← clear-bit (token, reg2) Store-memory (adr, token, Byte) | |
| | Note 1. An MDP exception might occur depending on the result of address calcul | ation. |
| [Format] | (1) Format VIII | |
| | (2) Format IX | |
| [Opcode] | 15 0.21 40 | |
| | (1) 10bbb111110RRRRR dddddddddddddddddd | |
| | 15 0.31 16 (2) rrrrr111111RRRRR 0000000011100100 | |
| [Flags] | CY — | |
| | ov — | |
| | S — | |
| | Z "1" if bit specified by operand = "0", "0" if bit specified by operand = "1". SAT — | |
| | | |



| [Description] | (1) Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Byte data is read from the generated address, then the bits indicated by the 3-bit bit number are cleared (0) and the data is written back to the original address. | | | | |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|--|
| | (2) Reads the word data of general-purpose register reg1 to generate a 32-bit address. Byte data is read from the generated address, the bits indicated by the lower three bits of reg2 are cleared (0), and the data is written back to the original address. | | | | |
| [Supplement] | The Z flag of PSW indicates the status of the specified bit (0 or 1) before this instruction is executed, and does not indicate the content of the specified bit after this instruction is executed. | | | | |
| | CAUTION | | | | |
| | This instruction provides an atomic guarantee aimed at exclusive control, and during the period between read and write operations, the target address is not affected by access due to any | | | | |

other cause.



| СМОУ | | Conditional move |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------|------------------|
| | | Conditional move |
| [Instruction format] | (1) CMOV cccc, reg1, reg2, reg3 | |
| | (2) CMOV cccc, imm5, reg2, reg3 | |
| [Operation] | (1) if conditions are satisfied then GR [reg3] ← GR [reg1] else GR [reg3] ← GR [reg2] | |
| | (2) if conditions are satisfied then GR [reg3] ← sign-extended (imm5) else GR [reg3] ← GR [reg2] | |
| [Format] | (1) Format XI | |
| | (2) Format XII | |
| [Opcode] | | |
| | 15 0 31 16 (1) rrrr111111RRRRR wwwww011001cccc0 | |
| | 15 0 31 16 | |
| | (2) rrrr111111iiii wwww011000cccc0 | |
| [Flags] | | |
| [1 1655] | CY — | |
| | S — | |
| | z — | |
| | SAT — | |
| | | |
| | | |



[Description]

(1) When the condition specified by condition code "cccc" is met, data in general-purpose register reg1 is transferred to general-purpose register reg3. When that condition is not met, data in general-purpose register reg2 is transferred to general-purpose register reg3. Specify one of the condition codes shown in the following table as "cccc".

| Condition Code | Name | Condition Formula | Condition Code | Name | Condition Formula |
|-------------------|-------|-------------------|-------------------|------|------------------------|
| 0000 | V | OV = 1 | 0100 | S/N | S = 1 |
| 1000 | NV | OV = 0 | 1100 | NS/P | S = 0 |
| 0001 | C/L | CY = 1 | 0101 | Т | Always (unconditional) |
| 1001 | NC/NL | CY = 0 | 1101 | SA | SAT = 1 |
| 0010 | Z | Z = 1 | 0110 | LT | (S xor OV) = 1 |
| 1010 | NZ | Z = 0 | 1110 | GE | (S xor OV) = 0 |
| 0011 | NH | (CY or Z) = 1 | 0111 | LE | ((S xor OV) or Z) = 1 |
| 1011 | Н | (CY or Z) = 0 | 1111 | GT | ((S xor OV) or Z) = 0 |

(2) When the condition specified by condition code "cccc" is met, 5-bit immediate data signextended to word-length is transferred to general-purpose register reg3. When that condition is not met, the data in general-purpose register reg2 is transferred to generalpurpose register reg3. Specify one of the condition codes shown in the following table as "cccc".

| Condition Code | Name | Condition Formula | Condition Code | Name | Condition Formula |
|-------------------|-------|-------------------|-------------------|------|------------------------|
| 0000 | V | OV = 1 | 0100 | S/N | S = 1 |
| 1000 | NV | OV = 0 | 1100 | NS/P | S = 0 |
| 0001 | C/L | CY = 1 | 0101 | Т | Always (unconditional) |
| 1001 | NC/NL | CY = 0 | 1101 | SA | SAT = 1 |
| 0010 | Z | Z = 1 | 0110 | LT | (S xor OV) = 1 |
| 1010 | NZ | Z = 0 | 1110 | GE | (S xor OV) = 0 |
| 0011 | NH | (CY or Z) = 1 | 0111 | LE | ((S xor OV) or Z) = 1 |
| 1011 | Н | (CY or Z) = 0 | 1111 | GT | ((S xor OV) or Z) = 0 |

[Supplement]

See the description of the SETF instruction.



| <arithmetic instruction=""></arithmetic> | | |
|------------------------------------------|-----------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| СМР | | Compare register/immediate (5-bit) |
| | | Compare |
| [Instruction format] | (1) CMP r | eg1. reg2 |
| [| (2) CMP in | mm5, reg2 |
| [Operation] | (1) result ← | – GR [reg2] – GR [reg1] |
| | (2) result \leftarrow | - GR [reg2] – sign-extend (imm5) |
| [Format] | (1) Format | I |
| | (2) Format | П |
| [Opcode] | 15 | 0 |
| | (1) rrrr | r001111RRRR |
| | (2) 15 (2) rrrr | 0 rr010011iiiii |
| [Flags] | | |
| [1.1.60] | CY | "1" if a borrow occurs from MSB; otherwise, "0". |
| | OV | "1" if overflow occurs; otherwise, "0". |
| | S | "1" if the operation result is negative; otherwise, "0". |
| | Z SAT | "1" if the operation result is "0"; otherwise, "0". — |
| [Description] | (1) Compar purpose perform register | res the word data of general-purpose register reg2 with the word data of general- e register reg1 and outputs the result through the PSW flags. Comparison is ned by subtracting the reg1 contents from the reg2 word data. General-purpose s reg1 and reg2 are not affected. |
| | (2) Compar sign-ext is perfo General | res the word data of general-purpose register reg2 with the 5-bit immediate data, tended to word length, and outputs the result through the PSW flags. Comparison rmed by subtracting the sign-extended immediate data from the reg2 word data. I-purpose register reg2 is not affected. |



| | | Return from CALLT |
|----------------------|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| CIREI | | |
| | | Return from subroutine call |
| [Instruction format] | CTRET | |
| [Operation] | PC ← CTPC | |
| | $PSW(4:0) \leftarrow$ | CTPSW(4:0) |
| [Format] | Format X | |
| [Opcode] | 15 | 031 16 |
| | 000001111 | 1100000 000000101000100 |
| [Flags] | | |
| | CY | Value read from CTPSW is set. |
| | OV | Value read from CTPSW is set. |
| | S | Value read from CTPSW is set. |
| | Z | Value read from CTPSW is set. |
| | SAT | Value read from CTPSW is set. |
| [Description] | Loads the returns from a | urn PC and PSW (the lower 5 bits) from the appropriate system register and a routine under CALLT instruction. The following steps are taken: |
| | (1) The return CTPSW. | rn PC and the return PSW (the lower 5 bits) are loaded from the CTPC and |
| | (2) The value the return | es are restored in PC and PSW (the lower 5 bits) and the control is transferred to n address. |
| | CAUTION | |
| | When the CT | RET instruction is executed, only the lower 5 bits of the PSW register are |



| | | | | | | Disable interrupt |
|----------------------|-------------------------------------|-------------------------------------------------|-------------------------------------------------------|-----------------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------------------|
| DI | | | | | Disable El level ma | skable exception |
| [Instruction format] | DI | | | | | |
| [Operation] | PSW.ID ∢ | — 1 (Disable | es EI level masl | cable interrupt) | | |
| [Format] | Format X | | | | | |
| [Opcode] | 15 000001: | 111110000 | 031 00 00000001 | 16 01100000 | | |
| | <u> </u> | | | | | |
| [Flags] | CY | _ | | | | |
| | OV | _ | | | | |
| | S | _ | | | | |
| | Z | — | | | | |
| | SAT | — | | | | |
| | ID | 1 | | | | |
| [Description] | Sets "1" t exception | o the ID flag s after the ex | g of the PSW to xecution of this | disable the acknow instruction. | vledgement of EI level | maskable |
| [Supplement] | Overwrite If the MC If the MC | e of flags in t TL.UIC bit I TL.UIC bit I | the PSW by this has been cleare has been set to | s instruction becon d to 0, this instruct 1, this instruction c | nes valid as of the next ion is a supervisor-leve can always be executed | instruction. I instruction. |



| DISPOSE |
|---------|
|---------|

Function dispose

Stack frame deletion

| [Instruction format] | (1) | DISPOSE imm5, list12 |
|----------------------|-----|--------------------------------------------------------------------------------------|
| | (2) | DISPOSE imm5, list12, [reg1] |
| [Operation] | (1) | $tmp \leftarrow sp + zero-extend (imm5 logically shift by 2)$ |
| | | foreach (all regs in list12) { adr \leftarrow tmp ^{*1, *2} |
| | | GR[reg in list12] ← Load-memory (adr, Word) |
| | | $tmp \leftarrow tmp + 4$ |
| | | } |
| | | sp ← tmp |
| | (2) | $tmp \leftarrow sp + zero-extend (imm5 logically shift by 2)$ |
| | | foreach (all regs in list12) { adr \leftarrow tmp ^{*1, *2} |
| | | GR[reg in list12] ← Load-memory (adr, Word) |
| | | $tmp \leftarrow tmp + 4$ |
| | | } |
| | | $PC \leftarrow GR[reg1]$ |
| | | $sp \leftarrow tmp$ |
| | | Note 1. An MDP exception might occur depending on the result of address calculation. |
| | | Note 2. When loading to memory, the lower 2 bits of adr are masked to 0. |
| [Format] | For | mat XIII |
| [Opcode] | | |
| - | | 15 0 31 16 |
| | (1) | 0000011001iiiiiL LLLLLLLL00000 |
| | | <u>15 0 31 16</u> |
| | (2) | 0000011001iiiiiLLLLLLLLLRRRRR |
| | RRF | RRR \neq 00000 (Do not specify r0 for reg1.) |

The values of LLLLLLLLLL are the corresponding bit values shown in register list "list12" (for example, the "L" at bit 21 of the opcode corresponds to the value of bit21 in list12). list12 is a 32-bit register list, defined as follows.

RENESAS

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|-----|
| r24 | r25 | r26 | r27 | r20 | r21 | r22 | r23 | r28 | r29 | r31 | _ | r30 |

Bits 31 to 21 and bit 0 correspond to general-purpose registers (r20 to r31), so that when any of these bits is set (1), it specifies a corresponding register operation as a processing target. For example, when r20 and r30 are specified, the values in list12 appear as shown below (register bits that do not correspond, i.e., bits 20 to 1 are set as "Don't care").

- When all of the register's non-corresponding bits are "0": $0800\ 0001_{\rm H}$
- When all of the register's non-corresponding bits are "1": $081F FFFF_H$

[Flags]

| CY | _ |
|-----|---|
| OV | — |
| S | _ |
| Z | _ |
| SAT | _ |

[Description]

- (1) Adds the 5-bit immediate data, logically left-shifted by 2 and zero-extended to word length, to sp; returns to general-purpose registers listed in list12 by loading the data from the address specified by sp and adds 4 to sp.
- (2) Adds the 5-bit immediate data, logically left-shifted by 2 and zero-extended to word length, to sp; returns to general-purpose registers listed in list12 by loading the data from the address specified by sp and adds 4 to sp; and transfers the control to the address specified by general-purpose register reg1.

[Supplement] General-purpose registers in list12 are loaded in descending order (r31, r30, ... r20). The imm5 restores a stack frame for automatic variables and temporary data. The lower 2 bits of the address specified by sp is always masked to "0" and aligned to the word boundary.

CAUTIONS

- 1. If an exception occurs while this instruction is being executed, execution of the instruction might be stopped after the read/write cycle and the register value write operation are completed, but sp will retain its original value from before the start of execution. The instruction will be executed again later, after a return from the exception.
- 2. For instruction format (2) DISPOSE imm5, list12, [reg1], do not specify r0 for reg1.



<Divide instruction>

| DIV | Divide word |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Division of (signed) word data |
| [Instruction format] | DIV reg1, reg2, reg3 |
| [Operation] | $GR [reg2] \leftarrow GR [reg2] \div GR [reg1]$ $GR [reg3] \leftarrow GR [reg2] \% GR [reg1]$ |
| [Format] | Format XI |
| [Opcode] | 15 031 16 rrrr111111RRRRR wwww01011000000 |
| [Flags] | CY—OV"1" if overflow occurs; otherwise, "0"S"1" if the operation result quotient is negative; otherwise, "0".Z"1" if the operation result quotient is "0"; otherwise, "0".SAT— |
| [Description] | Divides the word data of general-purpose register reg2 by the word data of general-purpose register reg1 and stores the quotient to general-purpose register reg2 with the remainder set to general-purpose register reg3. General-purpose register reg1 is not affected. When division by zero occurs, an overflow results and all operation results except for the OV flag are undefined. |
| [Supplement] | Overflow occurs when the maximum negative value $(8000\ 0000_{\rm H})$ is divided by -1 with the quotient = $8000\ 0000_{\rm H}$ and when the data is divided by 0 with quotient being undefined. If reg2 and reg3 are the same register, the remainder is stored in that register. When an exception occurs during the DIV instruction execution, the execution is aborted to process the exception. The execution resumes at the original instruction address upon returning from the exception. General-purpose register reg1 and general-purpose register reg2 retain their values prior to execution of this instruction. |
| | CAUTION |
| | If general-purpose registers reg2 and reg3 are specified as being the same register, the operation result quotient is not stored in reg2, so the flag is undefined. |



<Divide instruction>

| ЫЛН | Divide halfword |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Division of (signed) halfword data |
| [Instruction format] | (1) DIVH reg1, reg2 (2) DIVH reg1, reg2, reg3 |
| [Operation] | GR [reg2] ← GR [reg2] ÷ sign-extend (GR [reg1] (15:0)) GR [reg2] ← GR [reg2] ÷ sign-extend (GR [reg1] (15:0)) GR [reg3] ← GR [reg2] % sign-extend (GR [reg1] (15:0)) |
| [Format] | (1) Format I (2) Format XI |
| [Opcode] | $15 	 0$ (1) rrrr000010RRRR RRRR \neq 00000 (Do not specify r0 for reg1.) rrrrr \neq 00000 (Do not specify r0 for reg2.) $15 	 031 	 16$ (2) rrrr111111RRRR wwww0101000000 |
| [Flags] | CY—OV"1" if overflow occurs; otherwise, "0".S"1" if the operation result quotient is negative; otherwise, "0".Z"1" if the operation result quotient is "0"; otherwise, "0".SAT— |
| [Description] | Divides the word data of general-purpose register reg2 by the lower halfword data of general-purpose register reg1 and stores the quotient to general-purpose register reg2. General-purpose register reg1 is not affected. When division by zero occurs, an overflow results and all operation results except for the OV flag are undefined. Divides the word data of general-purpose register reg2 by the lower halfword data of general-purpose register reg1 and stores the quotient to general-purpose register reg2 with the remainder set to general-purpose register reg3. General-purpose register reg1 is not affected. When division by zero occurs, an overflow results and all operation by zero occurs, an overflow results and all operation results except for the OV flag are undefined. |

RENESAS

[Supplement] (1) The remainder is not stored. Overflow occurs when the maximum negative value (8000 0000_H) is divided by -1 with the quotient = 8000 0000_H and when the data is divided by 0 with quotient being undefined. When an exception occurs during the DIVH instruction execution, the execution is aborted to process the exception. General-purpose register reg1 and general-purpose register reg2 retain their values prior to execution of this instruction.
(2) Overflow occurs when the maximum negative value (8000 0000_H) is divided by -1 with the quotient = 8000 0000_H and when the data is divided by 0 with quotient being undefined. If reg2 and reg3 are the same register, the remainder is stored in that register. When an exception occurs during the DIVH instruction execution the execution is

When an exception occurs during the DIVH instruction execution, the execution is aborted to process the exception. The execution resumes at the original instruction address upon returning from the exception. General-purpose register reg1 and generalpurpose register reg2 retain their values prior to execution of this instruction.

CAUTIONS

- 1. If general-purpose registers reg2 and reg3 are specified as being the same register, the operation result quotient is not stored in reg2, so the flag is undefined.
- 2. Do not specify r0 as reg1 and reg2 for DIVH reg1 and reg2 in instruction format (1).



<Divide instruction>

| DIVHU | Divide halfword unsigned |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Division of (unsigned) halfword data |
| [Instruction format] | DIVHU reg1, reg2, reg3 |
| [Operation] | $GR [reg2] \leftarrow GR [reg2] \div zero-extend (GR [reg1] (15:0))$ $GR [reg3] \leftarrow GR [reg2] \% zero-extend (GR [reg1] (15:0))$ |
| [Format] | Format XI |
| [Opcode] | 15 031 16 rrrr111111RRRRR wwww01010000010 |
| [Flags] | CY—OV"1" if overflow occurs; otherwise, "0".S"1" when the operation result quotient word data is "1"; otherwise, "0"Z"1" if the operation result quotient is "0"; otherwise, "0".SAT— |
| [Description] | Divides the word data of general-purpose register reg2 by the lower halfword data of general- purpose register reg1 and stores the quotient to general-purpose register reg2 with the remainder set to general-purpose register reg3. General-purpose register reg1 is not affected. When division by zero occurs, an overflow results and all operation results except for the OV flag are undefined. |
| [Supplement] | Overflow occurs by division by zero (with the operation result being undefined). If reg2 and reg3 are the same register, the remainder is stored in that register. When an exception occurs during the DIVHU instruction execution, the execution is aborted to process the exception. The execution resumes at the original instruction address upon returning from the exception. General-purpose register reg1 and general-purpose register reg2 retain their values prior to execution of this instruction. |
| | CAUTION |
| | If general-purpose registers reg2 and reg3 are specified as being the same register, the operation result quotient is not stored in reg2, so the flag is undefined. |


<High-speed divide instructions>

| DIVQ | Divide word quickly |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Division of (signed) word data (variable steps) |
| [Instruction format] | DIVQ reg1, reg2, reg3 |
| [Operation] | $GR [reg2] \leftarrow GR [reg2] \div GR [reg1]$ $GR [reg3] \leftarrow GR [reg2] \% GR [reg1]$ |
| [Format] | Format XI |
| [Opcode] | 15 031 16 rrrr111111RRRRR wwww01011111100 |
| [Flags] | CY—OV"1" when overflow occurs; otherwise, "0".S"1" when operation result quotient is a negative value; otherwise, "0".Z"1" when operation result quotient is a "0"; otherwise, "0".SAT— |
| [Description] | Divides the word data in general-purpose register reg2 by the word data in general-purpose register reg1, stores the quotient in reg2, and stores the remainder in general-purpose register reg3. General-purpose register reg1 is not affected. The minimum number of steps required for division is determined from the values in reg1 and reg2, then this operation is executed. When division by zero occurs, an overflow results and all operation results except for the OV flag are undefined. |
| [Supplement] | Overflow occurs when the maximum negative value (8000 0000_H) is divided by -1 (with the quotient = 8000 0000_H) and when the data is divided by 0 with the quotient being undefined. If reg2 and reg3 are the same register, the remainder is stored in that register. When an exception occurs during execution of this instruction, the execution is aborted. After exception handling is completed, the execution resumes at the original instruction address when returning from the exception. General-purpose register reg1 and general-purpose register reg2 retain their values prior to execution of this instruction. The smaller the difference in the number of valid bits between reg1 and reg2, the smaller |
| | the number of execution cycles. In most cases, the number of instruction cycles is smaller than that of the ordinary division instruction. If data of 16-bit integer type is divided by another 16-bit integer type data, the difference in the number of valid bits is 15 or less, and the operation is completed within 20 cycles. |



CAUTIONS

- 1. If general-purpose registers reg2 and reg3 are specified as being the same register, the operation result quotient is not stored in reg2, so the flag is undefined.
- 2. For the accurate number of execution cycles, see the appendix.
- 3. If the number of execution cycles must always be constant to guarantee real-time features, use the ordinary division instruction.



<High-speed divide instructions>

| DIVQU | Divide word unsigned quickly |
|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Division of (unsigned) word data (variable steps) |
| [Instruction format] | DIVQU reg1, reg2, reg3 |
| [Operation] | $GR [reg2] \leftarrow GR [reg2] \div GR [reg1]$ $GR [reg3] \leftarrow GR [reg2] \% GR [reg1]$ |
| [Format] | Format XI |
| [Opcode] | 15 031 16 rrrr111111RRRRR wwww0101111110 |
| [Flags] | CY—OV"1" when overflow occurs; otherwise, "0".S"1" when operation result quotient is a negative value; otherwise, "0".Z"1" when operation result quotient is a "0"; otherwise, "0".SAT— |
| [Description] | Divides the word data in general-purpose register reg2 by the word data in general-purpose register reg1, stores the quotient in reg2, and stores the remainder in general-purpose register reg3. General-purpose register reg1 is not affected. The minimum number of steps required for division is determined from the values in reg1 and reg2, then this operation is executed. When division by zero occurs, an overflow results and all operation results except for the OV flag are undefined. |
| [Supplement] | (1) An overflow occurs when there is division by zero (the operation result is undefined). If reg2 and reg3 are the same register, the remainder is stored in that register. When an exception occurs during execution of this instruction, the execution is aborted. After exception handling is completed, using the return address as this instruction's start address, the execution resumes when returning from the exception. General-purpose register reg1 and general-purpose register reg2 retain their values prior to execution of this instruction. |
| | (2) The smaller the difference in the number of valid bits between reg1 and reg2, the smaller the number of execution cycles. In most cases, the number of instruction cycles is smaller than that of the ordinary division instruction. If data of 16-bit integer type is divided by another 16-bit integer type data, the difference in the number of valid bits is 15 or less, and the operation is completed within 20 cycles. |



CAUTIONS

- 1. If general-purpose registers reg2 and reg3 are specified as being the same register, the operation result quotient is not stored in reg2, so the flag is undefined.
- 2. For the accurate number of execution cycles, see the appendix.
- 3. If the number of execution cycles must always be constant to guarantee real-time features, use the ordinary division instruction.



<Divide instruction>

| DIVU | Divide word unsigned |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Division of (unsigned) word data |
| [Instruction format] | DIVU reg1, reg2, reg3 |
| [Operation] | $GR [reg2] \leftarrow GR [reg2] \div GR [reg1]$ $GR [reg3] \leftarrow GR [reg2] \% GR [reg1]$ |
| [Format] | Format XI |
| [Opcode] | 15 031 16 rrrr111111RRRRR wwww01011000010 |
| [Flags] | CY—OV"1" if overflow occurs; otherwise, "0".S"1" when operation result quotient word data MSB is "1"; otherwise, "0".Z"1" if the operation result quotient is "0"; otherwise, "0".SAT— |
| [Description] | Divides the word data of general-purpose register reg2 by the word data of general-purpose register reg1 and stores the quotient to general-purpose register reg2 with the remainder set to general-purpose register reg3. General-purpose register reg1 is not affected. When division by zero occurs, an overflow results and all operation results except for the OV flag are undefined. |
| [Supplement] | When an exception occurs during the DIVU instruction execution, the execution is aborted to process the exception.If reg2 and reg3 are the same register, the remainder is stored in that register.The execution resumes at the original instruction address upon returning from the exception.General-purpose register reg1 and general-purpose register reg2 retain their values prior to execution of this instruction. |
| | CAUTION |
| | If general-purpose registers reg2 and reg3 are specified as being the same register, the operation result quotient is not stored in reg2, so the flag is undefined. |



| | | Enabl | e interrupt |
|----------------------|---------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| | | Enable El level maskable | exception |
| [Instruction format] | EI | | |
| [Operation] | PSW.ID ← | 0 (enables EI level maskable exception) | |
| [Format] | Format X | | |
| [Opcode] | 15 1000011 | 031 16 111100000 000000101100000 | |
| [Flags] | CY OV S Z SAT ID | | |
| [Description] | Clears the exceptions | ID flag of the PSW to "0" and enables the acknowledgement of maskable starting the next instruction. | |
| [Supplement] | If the MC If the MC | CL.UIC bit has been cleared to 0, this instruction is a supervisor-level instr CL.UIC bit has been set to 1, this instruction can always be executed. | ruction. |



| FIDET | | Return from trap or interrupt |
|----------------------|-------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| EIREI | | |
| | | Return from EL level exception |
| [Instruction format] | EIRET | |
| [Operation] | PC ← EI | PC |
| | PSW ← I | EIPSW |
| [Format] | Format X | |
| [Opcode] | 15 | 031 16 |
| | 000001 | 1111100000 000000101001000 |
| | | |
| [Flags] | CY | Value read from EIPSW is set |
| | OV | Value read from EIPSW is set |
| | S | Value read from EIPSW is set |
| | Z | Value read from EIPSW is set |
| | SAT | Value read from EIPSW is set |
| [Description] | Returns e EIPC and If EP = 0, the ISPR | xecution from an EI level exception. The return PC and PSW are loaded from the EIPSW registers and set in the PC and PSW, and control is passed. it means that interrupt (EIINT <i>n</i>) processing has finished, so the corresponding bit of register is cleared. |
| [Supplement] | This instr | uction is a supervisor-level instruction. |



| | | Return from trap or interrupt | |
|----------------------|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| FEREI | | | |
| | | Return from FE level exception | |
| [Instruction format] | FEDET | | |
| [Instruction format] | FEKE1 | | |
| [Operation] | PC ← FEP | C | |
| | DSW ∠ FE | DCW/ | |
| | 150 - 11 | 1 5 W | |
| [Format] | Format X | | |
| | | | |
| [Opcode] | 15 | 031 16 | |
| | 00000111 | .11100000 000000101001010 | |
| | | | |
| | | | |
| [Flags] | CY | Value read from EEPSW is set | |
| | OV | Value read from FEPSW is set | |
| | S | Value read from FEPSW is set | |
| | Z | Value read from FEPSW is set | |
| | SAT | Value read from FEPSW is set | |
| [Description] | Returns exe | ecution from an FE level exception. The return PC and PSW are loaded from the | |
| | FEPC and I | FEPSW registers and set in the PC and PSW, and control is passed. | |
| [Supplement] | This instruc | ction is a supervisor-level instruction. | |
| | CAUTION | | |
| | The FERET operating si instruction t with the UN hardware fu instruction i the FERET | instruction can also be used as a hazard barrier instruction when the CPU's tatus (PSW) is changed by a control program such as the OS. Use the FERET to clarify the program blocks on which to effect the hardware function associated I bit in the PSW when these bits are changed to accord with the mounted CPU. The unction that operates in accordance with the PSW value updated by the FERET s guaranteed to be effected from the instruction indicated by the return address of instruction. | |



| FFTRAP | | | FE-level Trap |
|----------------------|---------------------|------------------------------------------------|-----------------------------|
| | | | FE level software exception |
| [Instruction format] | FETRAP | vector4 | |
| [Operation] | $FEPC \leftarrow I$ | PC + 2 (return PC) | |
| | $FEPSW \leftarrow$ | - PSW | |
| | FEIC \leftarrow e | xception cause code ¹ | |
| | PSW.UM | $\leftarrow 0$ | |
| | PSW.NP ↔ | - 1 | |
| | PSW.EP ← | - 1 | |
| | PSW.ID ← | -] | |
| | $PC \leftarrow exc$ | eption handler address – | |
| | Note | 1. See Table 4.1, Exception Cause List. | |
| | Note | 2. See Section 4.4, Exception Handler Address. | |
| | | | |
| [Format] | Format I | | |
| [Opcode] | | | |
| - | 15 | 0 | |
| | 0vvvv00 | 001000000 | |
| | Where vv | vv is vector4. | |
| | Do not set | $0_{\rm H}$ to vector4 (vvvv \neq 0000). | |
| | | | |
| [Flags] | | | |
| | CY | — | |
| | OV | — | |
| | S | — | |
| | Z | _ | |
| | SAT | _ | |



[Description]

Saves the contents of the return PC (address of the instruction next to the FETRAP instruction) and the current contents of the PSW to FEPC and FEPSW, respectively, stores the exception cause code in the FEIC register, and updates the PSW according to the exception causes listed in **Table 4.1**. Execution then branches to the exception handler address and exception handling is started.

Table 7.6 shows the correspondence between vector4 and exception cause codes andexception handler address offset. Exception handler addresses are calculated based on theoffset addresses listed in Table 7.6. For details, see Section 4.4, Exception HandlerAddress.

Table 7.6 Correspondence between vector4 and Exception Cause Codes and Exception Handler Address Offset

| vector4 | Exception Cause Code | Offset Address |
|----------------|------------------------|-----------------|
| 0 _H | | Not specifiable |
| 1 _H | 0000 0031 _H | 30 _H |
| 2 _H | 0000 0032 _H | |
| | | |
| F _H | 0000 003F _H | |



| HALT | Halt |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | HALT |
| [Operation] | Places the CPU core in the HALT state. |
| [Format] | Format X |
| [Opcode] | 15 031 16 0000011111100000 000000100100000 |
| [Flags] | CY — OV — S — Z — SAT — |
| [Description] | Places the CPU core that executed the HALT instruction in the HALT state.Occurrence of the HALT state release request will return the system to normal execution status.If an exception is acknowledged while the system is in HALT state, the return PC of that exception is the PC of the instruction that follows the HALT instruction.The HALT state is released under the following condition. |
| | A terminating exception occurs Even if the conditions for acknowledging the above exceptions are not satisfied (due to the ID or NP value), as long as a HALT mode release request exists, HALT state is released (for example, even if PSW.ID = 1, HALT state is released when INT0 occurs). Note, however, that the HALT mode will not be released if terminating exceptions are masked by the following mask settings, which are defined individually for each function: Terminating exceptions are masked by an interrupt channel mask setting specified by the interrupt controller*1. Terminating exceptions are masked by a mask setting specified by using the floating-point operation exception enable bit. Terminating exceptions are masked by a mask setting defined by a hardware function other than the above. |

Note 1. This does not include masking specified by the ISPR and PMR registers.

[Supplement]

This instruction is a supervisor-level instruction.



<Data manipulation instructions>

| нѕн | | Halfword swap halfword |
|----------------------|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Halfword swap of halfword data |
| [Instruction format] | HSH reg | 2, reg3 |
| [Operation] | GR [reg3 | $] \leftarrow GR [reg2]$ |
| [Format] | Format X | II |
| [Opcode] | 15 rrrrl: | 031 16 1111100000 wwwww01101000110 |
| [Flags] | CY OV S Z SAT | "1" if the lower halfword of the operation result is "0"; otherwise, "0". "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the lower halfword of the operation result is "0"; otherwise, "0". |
| [Description] | Stores the the flag ju | content of general-purpose register reg2 in general-purpose register reg3, and stores adgment result in PSW. |



<Data manipulation instructions>

| HSW | | Halfword swap word |
|----------------------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Halfword swap of word data |
| [Instruction format] | HSW reg | g2, reg3 |
| [Operation] | GR [reg3 | GR [reg2] (15:0) GR [reg2] (31:16) |
| [Format] | Format X | |
| [Opcode] | 15 rrrrrl | 031 16 1111100000 wwww01101000100 |
| [Flags] | CY OV S Z SAT | "1" when there is at least one halfword of zero in the word data of the operation result; otherwise; "0". 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if operation result word data is "0"; otherwise, "0". |
| [Description] | Executes | endian swap. |



<Branch instruction>

| | | Jump and register link |
|----------------------|--------------------------------------------------------------------------------|--------------------------|
| | | Branch and register link |
| [Instruction format] | (1) JARL disp22, reg2 | |
| | (2) JARL disp32, reg1 | |
| | (3) JARL [reg1], reg3 | |
| [Operation] | (1) GR [reg2] \leftarrow PC + 4 PC \leftarrow PC + sign-extend (disp22) | |
| | (2) GR [reg1] \leftarrow PC + 6 PC \leftarrow PC + disp32 | |
| | (3) $GR[reg3] \leftarrow PC + 4$ $PC \leftarrow GR[reg1]$ | |
| [Format] | (1) Format V | |
| | (2) Format VI | |
| | (3) Format XI | |
| [Opcode] | | |
| | 150 3116(1)rrrr11110ddddddddddddddddddddddddddddddd | |
| | ddddddddddddddd is the higher 21 bits of disp22. | |
| | $rrrrr \neq 00000$ (Do not specify r0 for reg2.) | |

| | 15 C | 16 | 47 32 | |
|-----|------------------|---------------|------------------|--|
| (2) | 00000010111RRRRR | ddddddddddddd | סססססססססססססססס | |

DDDDDDDDDDDDDDDDddddddddddd is the higher 31 bits of disp32. RRRRR $\neq 00000$ (Do not specify r0 for reg1.)

WWWWW \neq 00000 (Do not specify r0 for reg3.)



[Flags]

| CY | _ |
|-----|---|
| OV | — |
| S | _ |
| Z | _ |
| SAT | _ |

[Description]

- (1) Saves the current PC value + 4 in general-purpose register reg2, adds the 22-bit displacement data, sign-extended to word length, to PC; stores the value in and transfers the control to PC. Bit 0 of the 22-bit displacement is masked to "0".
- (2) Saves the current PC value + 6 in general-purpose register reg1, adds the 32-bit displacement data to PC and stores the value in and transfers the control to PC. Bit 0 of the 32-bit displacement is masked to "0".
- (3) Stores the current PC value + 4 in reg3, specifies the contents of reg1 for the PC value, and then transfers the control.

[Supplement] The current PC value used for calculation is the address of the first byte of this instruction itself. The jump destination is this instruction with the displacement value = 0. JARL instruction corresponds to the call function of the subroutine control instruction, and saves the return PC address in either reg1 or reg2. JMP instruction corresponds to the return function of the subroutine control instruction, and can be used to specify general-purpose register containing the return address as reg1 to the return PC.

CAUTION

Do not specify r0 for the general-purpose register reg2 in the instruction format (1) JARL disp22, reg2.

Do not specify r0 for the general-purpose register reg1 in the instruction format (2) JARL disp32, reg1.

Do not specify r0 for the general-purpose register reg3 in the instruction format (3) JARL [reg1], reg3.



<Branch instruction>

| JMP | Jump register |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Unconditional branch (register relative) |
| [Instruction format] | (1) JMP [reg1] (2) JMP disp32 [reg1] |
| [Operation] | (1) $PC \leftarrow GR [reg1]$ (2) $PC \leftarrow GR [reg1] + disp32$ |
| [Format] | Format I Format VI |
| [Opcode] | (1) $\begin{bmatrix} 15 & 0 \\ 0000000011RRRR \end{bmatrix}$ (2) $\begin{bmatrix} 15 & 0.31 & 16.47 & 32 \\ 00000110111RRRR & dddddddddddddddddddddddddddddd$ |
| [Flags] | CY — OV — S — Z — SAT — |
| [Description] | Transfers the control to the address specified by general-purpose register reg1. Bit 0 of the address is masked to "0". Adds the 32-bit displacement to general-purpose register reg1, and transfers the control to the resulting address. Bit 0 of the address is masked to "0". |
| [Supplement] | Using this instruction as the subroutine control instruction requires the return PC to be specified by general-purpose register reg1. |



<Branch instruction>

| JR | Jump relative |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Unconditional branch (PC relative) |
| [Instruction format] | (1) JR disp22 (2) JR disp32 |
| [Operation] | (1) $PC \leftarrow PC + sign-extend (disp22)$ (2) $PC \leftarrow PC + disp32$ |
| [Format] | (1) Format V (2) Format VI |
| [Opcode] | (1) 15 031 16 (1) 0000011110ddddd ddddddddddddddddddddd |
| | 150.3116.4732(2)000001011100000ddddddddddddddddddddddddddddddd |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the 22-bit displacement data, sign-extended to word length, to the current PC and stores the value in and transfers the control to PC. Bit 0 of the 22-bit displacement is masked to "0". |
| | (2) Adds the 32-bit displacement data to the current PC and stores the value in PC and transfers the control to PC. Bit 0 of the 32-bit displacement is masked to "0". |
| [Supplement] | The current PC value used for calculation is the address of the first byte of this instruction itself. The displacement value being "0" signifies that the branch destination is the instruction itself. |

RENESAS

<Load instruction>

| LD.B | Load byte |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Load of (signed) byte data |
| [Instruction format] | (1) LD.B disp16 [reg1], reg2 (2) LD.B disp23 [reg1], reg3 |
| [Operation] | (1) adr ← GR [reg1] + sign-extend (disp16)*1 GR [reg2] ← sign-extend (Load-memory (adr, Byte)) |
| | (2) adr ← GR [reg1] + sign-extend (disp23)*1 GR [reg3] ← sign-extend (Load-memory (adr, Byte)) |
| | Note 1. An MDP exception might occur depending on the result of address calculation. |
| [Format] | Format VII Format XIV |
| [Opcode] | 15 0 31 16 (1) rrrr111000RRRR dddddddddddddddddd |
| | (2) 00000111100RRRR wwwwddddddd0101 DDDDDDDDDDDDDDDDD |
| | Where RRRRR = reg1, wwwww = reg3. ddddddd is the lower side bits 6 to 1 of disp23. DDDDDDDDDDDDDDDDD is the higher 16 bits of disp23. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | (1) Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign extended to word length, to generate a 32-bit address. Byte data is read from the generated address, sign-extended to word length, and stored in general-purpose register reg2. |
| | (2) Adds the word data of general-purpose register reg1 to the 23-bit displacement data, sign |

extended to word length, to generate a 32-bit address. Byte data is read from the generated address, sign-extended to word length, and stored in general-purpose register reg3.

<Load instruction>

| | | | | Load byte unsigne |
|----------------------|------------------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------|-----------------------------------|
| | | | | Load of (unsigned) byte dat |
| [Instruction format] | (1) LD.E (2) LD.E | U disp16 [reg1] , re U disp23 [reg1] , re | eg2 eg3 | |
| [Operation] | (1) adr ← GR [1 | - GR [reg1] + sign-€ reg2] ← zero-extend | extend (disp16) ^{*1} l (Load-memory (adr, By | te)) |
| | (2) adr ← GR [| - GR [reg1] + sign-€ reg3] ← zero-extenc | extend (disp23) ^{*1} l (Load-memory (adr, By | te)) |
| | Note | 1. An MDP exception | might occur depending on th | ne result of address calculation. |
| [Format] | (1) Form (2) Form | at VII at XIV | | |
| [Opcode] | (1) rr ddddddd rrrrr≠ | rrr11110bRRRRF dddddddd is the h 00000 (Do not speci | igher 15 bits of disp16, a | nd b is bit 0 of disp16. |
| | 15 | (|) 31 16 | § 47 32 |
| | (2) 00 | 000111101RRRRF | wwwwwddddddd0101 | סססססססססססססס |
| | Where RR ddddddd DDDDDDD | RRR = reg1, wwww i is the lower 7 bits o DDDDDDDDD is the | w = reg3. of disp23. higher 16 bits of disp23. | |
| [Flags] | CY OV | | | |
| | S Z SAT | _ _ _ | | |



[Description]

- (1) Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Byte data is read from the generated address, zero-extended to word length, and stored in general-purpose register reg2.
- (2) Adds the word data of general-purpose register reg1 to the 23-bit displacement data, sign-extended to word length, to generate a 32-bit address. Byte data is read from the generated address, zero-extended to word length, and stored in general-purpose register reg3.

CAUTION

Do not specify r0 for reg2.



<Load instruction>

| LD.DW | | | | Load Double Word |
|----------------------|------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| | | | | Load of doubleword data |
| [Instruction format] | LD.DW disp2 | 3[reg1], reg3 | | |
| [Operation] | adr ← GR [reg data ← Load-n GR [reg3 + 1] | g1] + sign-extend (disp2 nemory (adr, Double-w ∥ GR [reg3] ← data | 23)* 1 vord) | |
| | Note 1. A | n MAE or MDP exception | might occur depending on th | ne result of address calculation. |
| [Format] | Format XIV | | | |
| [Opcode] | 15 0000011110 | 0 31)1RRRRR wwwwdddo | 16 47 ddd01001 DDDDDDDI | 32 DDDDDDDD |
| | Where RRRRR dddddd is the DDDDDDDDDD | R = reg1, wwwww = reg lower side bits 6 to 1 o DDDDDD is the higher | 3. of disp23. 16 bits of disp23. | |
| [Flags] | CY OV S Z SAT | | | |
| [Description] | Generates a 32 length to the w generated 32-b the higher 32 b | -bit address by adding ord data of general-pur it address and the lowe bits in reg3 + 1. | a 23-bit displacement val pose register reg1. Doub r 32 bits are stored in gen | lue sign-extended to word leword data is read from the eral-purpose register reg3, and |
| [Supplement] | reg3 must be a | n even-numbered regis | ter. | |



<Load instruction>

| | Load halfword |
|----------------------|---------------------------------------------------------------------------------------------|
| LD.H | Load of (unsigned) halfword data |
| | |
| [Instruction format] | (1) LD.H disp16 [reg1], reg2 (2) LD.H disp22 [$m=1$], $m=2$ |
| | (2) LD.H disp25 [reg1], reg5 |
| [Operation] | (1) $adr \leftarrow GR [reg1] + sign-extend (disp16)^{*1}$ |
| | $GR [reg2] \leftarrow sign-extend (Load-memory (adr, Halfword))$ |
| | (2) $adr \leftarrow GR [reg1] + sign-extend (disp23)^{*1}$ |
| | $GR [reg3] \leftarrow sign-extend (Load-memory (adr, Halfword))$ |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | (1) Format VII |
| | (2) Format XIV |
| [Opcode] | |
| | <u>15 0.31 16</u> |
| | (1) rrrr111001RRRRR dddddddddddddddd |
| | Where dddddddddddd is the higher 15 bits of disp16. |
| | |
| | <u>15 0 31 16 47 32</u> |
| | (2) 00000111100RRRRR www.wdddddd00111 DDDDDDDDDDDDDDD |
| | Where $RRRR = reg1$, wwwww = reg3. |
| | dddddd is the lower side bits 6 to 1 of disp23. |
| | DDDDDDDDDDDDD is the higher 16 bits of disp23. |
| | |
| [Flags] | CY — |
| | OV — |
| | s — |
| | Z — |
| | SAT — |



[Description]

- (1) Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Halfword data is read from this 32-bit address, sign-extended to word length, and stored in general-purpose register reg2.
- (2) Adds the word data of general-purpose register reg1 to the 23-bit displacement data, sign-extended to word length, to generate a 32-bit address. Halfword data is read from this 32-bit address, sign-extended to word length, and stored in general-purpose register reg3.



<Load instruction>

| прнп | Load halfword unsigned |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------|
| | Load of (signed) halfword data |
| [Instruction format] | (1) LD.HU disp16 [reg1], reg2 (2) LD.HU disp23 [reg1], reg3 |
| [Operation] | (1) adr ← GR [reg1] + sign-extend (disp16)*1 GR [reg2] ← zero-extend (Load-memory (adr, Halfword)) |
| | (2) adr ← GR [reg1] + sign-extend (disp23)*1 GR [reg3] ← zero-extend (Load-memory (adr, Halfword)) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | (1) Format VII (2) Format XIV |
| [Opcode] | (1) $\begin{array}{c} 15 & 0.31 & 16 \\ rrrrr111111RRRRR & dddddddddddddddddddddddddd$ |
| | (2) $15 	 0.31 	 16.47 	 32$ $00000111101RRRR wwwwdddddd00111 DDDDDDDDDDDDDDDDDDDDDDDDDDDD$ |
| | dddddd is the lower side bits 6 to1 of disp23. DDDDDDDDDDDDDDDD is the higher 16 bits of disp23. |
| [Flags] | CY – OV – S – Z – |
| | SAT — |



[Description]

- Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Halfword data is read from this 32-bit address, zero-extended to word length, and stored in general-purpose register reg2.
- (2) Adds the word data of general-purpose register reg1 to the 23-bit displacement data, sign-extended to word length, to generate a 32-bit address. Halfword data is read from this address, zero-extended to word length, and stored in general-purpose register reg3.

CAUTION

Do not specify r0 for reg2.



<Load instruction>

| LD.W | Load word |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| | Load of word data |
| [Instruction format] | (1) LD.W disp16 [reg1], reg2 (2) LD.W disp23 [reg1], reg3 |
| [Operation] | (1) adr ← GR [reg1] + sign-extend (disp16)*1 GR [reg2] ← Load-memory (adr, Word) |
| | (2) adr ← GR [reg1] + sign-extend (disp23)*1 GR [reg3] ← Load-memory (adr, Word) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format VII Format XIV |
| [Opcode] | 15 031 16 (1) rrrr111001RRRRR dddddddddddddddd |
| | Where dddddddddddd is the higher 15 bits of disp16. |
| | 15 0 31 16 47 32 (2) 00000111100RRRRR www.wdddddd01001 DDDDDDDDDDDDDDDDDDDDDDD |
| | Where RRRRR = reg1, wwwww = reg3. dddddd is the lower side bits 6 to 1 of disp23. DDDDDDDDDDDDDDDDD is the higher 16 bits of disp23. |
| [Flags] | CY – OV – S – Z – SAT – |



[Description]

- Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Word data is read from this 32-bit address, and stored in general-purpose register reg2.
- (2) Adds the word data of general-purpose register reg1 to the 23-bit displacement data, sign-extended to word length, to generate a 32-bit address. Word data is read from this address, and stored in general-purpose register reg3.



| LDL.W | Load Linked |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Load to start atomic word data manipulation |
| [Instruction format] | LDL.W [reg1], reg3 |
| [Operation] | adr \leftarrow GR[reg1] ^{*1} GR[reg3] \leftarrow Load-memory (adr, Word) LLbit $\leftarrow 1^{*2}$ |
| | Note 1. An MAE, MDP, or DTLBE exception might occur depending on the result of address calculation. |
| | Note 2. The result of an interrupt or exception, or the execution of a CLL, EIRET, or FERET instruction is LLbit ← 0. |
| [Format] | Format VII |
| [Opcode] | 15 031 16 00000111111RRRRR wwww01101111000 |
| [Flags] | CY — OV — S — Z — SAT — |
| [Description] | In order to perform an atomic read-modify-write operation, word data is read from the memory and stored in general-purpose register reg3. A link is then generated corresponding to the address range that includes the specified address. Subsequently, if a specific condition is satisfied before an STC.W instruction is executed for this LDL.W instruction, the link will be deleted. If an STC.W instruction is executed after the link has been deleted, STC.W execution will fail. If an STC.W instruction is executed while the link is still available, STC.W execution will succeed. The link is also deleted in this case. The LDL.W and STC.W instructions can be used to accurately update the memory in a multi core system. |
| [Supplement] | Use the LDL.W and STC.W instructions instead of the CAXI instruction if an atomic guarantee is required when updating the memory in a multi-core system. |



| | Load to system register |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LDJK | Load to system register |
| [Instruction format] | LDSR reg2, regID, selID LDSR reg2, regID |
| [Operation] | SR [regID, selID] \leftarrow GR [reg2] ^{*1} |
| | Note 1. An exception might occur depending on the access permission. For details, see Section 2.5.3, Register Updating. |
| [Format] | Format IX |
| [Opcode] | 15 0 31 16 rrrrr111111RRRR sssss00000100000 rrrrr: regID, sssss: selID, RRRR: reg2 |
| [Flags] | CY OV S Z SAT |
| [Description] | Loads the word data of general-purpose register reg2 to the system register specified by the system register number and group number (regID, selID). General-purpose register reg2 is not affected. If selID is omitted, it is assumed that selID is 0. |
| [Supplement] | A PIE or UCPOP exception might occur as a result of executing this instruction, depending on the combination of CPU operating mode and system register to be accessed. For details, see Section 2.5.3, Register Updating . CAUTIONS |
| | 1. In this instruction, general-purpose register reg2 is used as the source register, but, for mnemonic description convenience, the general-purpose register reg1 field is used in the opcode. The meanings of the register specifications in the mnemonic descriptions and opcode therefore differ from those of other instructions. |
| | The system register number or group number is a unique number used to identify each system register. How to access undefined registers is described in Section 2.5.4, Accessing Undefined Registers, but accessing undefined registers is not recommended. |



<Loop instruction>

| LOOP | | Loop Loop | |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| [Instruction format] | LOOP re | g1,disp16 | |
| [Operation] | $GR[reg1] \leftarrow GR[reg1] + (-1)^{*1}$ | | |
| | if $(GR[reg1] != 0)$ then $PC \leftarrow PC - zero-extend (disp16)$ | | |
| | Note | -1 (0xFFFFFFF) is added. The carry flag is updated in the same way as when the ADD instruction is executed. | |
| [Format] | Format VII | | |
| [Opcode] | 15 000001 Where d | 0.31 16 10111RRRRR dddddddddddddddd ddddddddddd is the higher 15 bits of disp16. | |
| [F]ags] | | | |
| [1 1023] | CY | "1" if a carry occurs from MSB in the reg1 operation; otherwise, "0". | |
| | OV | "1" if an overflow occurs in the reg1 operation; otherwise, "0". | |
| | S | "1" if reg1 is negative; otherwise, "0". | |
| | Z | "1" if reg1 is 0; otherwise, "0". | |
| | SAT | _ | |
| [Description] | Updates the general-purpose register reg1 by adding -1 from its contents. If the contents after this update are not 0, the following processing is performed. If the contents are 0, the system continues to the next instruction. | | |
| | • The result of logically shifting the 15-bit immediate data 1 bit to the left and zero- extending it to word length is subtracted from the current PC value, and then the control is transferred. | | |
| | • -1 (0 the s | 0xFFFFFFFF) is added to general-purpose register reg1. The carry flag is updated in ame way as when the ADD instruction, not the SUB instruction, is executed. | |



[Supplement]

"0" is implicitly used for bit 0 of the 16-bit displacement. Note that, because the current PC value used for calculation is the address of the first byte of this instruction, if the displacement value is 0, the branch destination is this instruction.

CAUTION

Do not specify r0 for reg1.



<Multiply-accumulate instruction>

| MAC | Multiply and add word | | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| | Multiply-accumulate for (signed) word data | | |
| [Instruction format] | MAC reg1, reg2, reg3, reg4 | | |
| [Operation] | $GR [reg4+1] \parallel GR [reg4] \leftarrow GR [reg2] \times GR [reg1] + GR [reg3+1] \parallel GR [reg3]$ | | |
| [Format] | Format XI | | |
| [Opcode] | 15 0.31 16 rrrrr111111RRRRR wwww0011110mmmm0 | | |
| [Flags] | CY – OV – S – Z – SAT – | | |
| [Description] | Multiplies the word data in general-purpose register reg2 by the word data in general-purpose register reg1, then adds the result (64-bit data) to 64-bit data consisting of the lower 32 bits of general-purpose register reg3 and the data in general-purpose register reg3+1 (for example, this would be "r7" if the reg3 value is r6 and "1" is added) as the higher 32 bits. Of the result (64-bit data), the higher 32 bits are stored in general-purpose register reg4+1 and the lower 32 bits are stored in general-purpose register reg4. The contents of general-purpose registers reg1 and reg2 are handled as 32-bit signed integers. This has no effect on general-purpose register reg1, reg2, reg3, or reg3+1. | | |

CAUTION

General-purpose registers that can be specified as reg3 or reg4 must be an even-numbered register (r0, r2, r4, ..., r30). The result is undefined if an odd-numbered register (r1, r3, ..., r31) is specified.



<Multiply-accumulate instruction>

| MACU | Multiply and add word unsigned | | | | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|--|
| | Multiply-accumulate for (unsigned) word data | | | | |
| [Instruction format] | MACU reg1, reg2, reg3, reg4 | | | | |
| [Operation] | $GR [reg4+1] \parallel GR [reg4] \leftarrow GR [reg2] \times GR [reg1] + GR [reg3+1] \parallel GR [reg3]$ | | | | |
| [Format] | Format XI | | | | |
| [Opcode] | 15 0 31 16 rrrrr111111RRRRR wwww0011111mmmm0 | | | | |
| [Flags] | CY OV S Z SAT | | | | |
| [Description] | Multiplies the word data in general-purpose register reg2 by the word data in general-purpose register reg1, then adds the result (64-bit data) to 64-bit data consisting of the lower 32 bits of general-purpose register reg3 and the data in general-purpose register reg3+1 (for example, this would be "r7" if the reg3 value is r6 and "1" is added) as the higher 32 bits. Of the result (64-bit data), the higher 32 bits are stored in general-purpose register reg4+1 and the lower 32 bits are stored in general-purpose register reg4. The contents of general-purpose registers reg1 and reg2 are handled as 32-bit signed integers. This has no effect on general-purpose register reg1, reg2, reg3, or reg3+1. | | | | |

CAUTION

General-purpose registers that can be specified as reg3 or reg4 must be an even-numbered register (r0, r2, r4, ..., r30). The result is undefined if an odd-numbered register (r1, r3, ..., r31) is specified.



<Arithmetic instruction>

| MOV | Move register/immediate (5-bit) /immediate (32-bit |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Data transfe |
| [Instruction format] | (1) MOV reg1, reg2 |
| | (2) MOV imm5, reg2 |
| | (3) MOV imm32, reg1 |
| [Operation] | (1) GR [reg2] \leftarrow GR [reg1] |
| | (2) GR [reg2] \leftarrow sign-extend (imm5) |
| | (3) GR [reg1] \leftarrow imm32 |
| [Format] | (1) Format I |
| | (2) Format II |
| | (3) Format VI |
| [Opcode] | $15 	 0 (1) 	 rrrr00000RRRR rrrr \neq 00000 (Do not specify r0 for reg2.) (2) 	 15 	 0 (2) 	 rrrrr010000iiiii) rrrrr \neq 00000 (Do not specify r0 for reg2.) 15 	 031 	 1647 	 32 (3) 	 10000110001RRRR 	 iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii$ |
| [Flags] | CY — OV — S — Z — SAT — |



[Description]

- (1) Copies and transfers the word data of general-purpose register reg1 to general-purpose register reg2. General-purpose register reg1 is not affected.
- (2) Copies and transfers the 5-bit immediate data, sign-extended to word length, to generalpurpose register reg2.
- (3) Copies and transfers the 32-bit immediate data to general-purpose register reg1.

CAUTION

Do not specify r0 as reg2 in MOV reg1, reg2 for instruction format (1) or in MOV imm5, reg2 for instruction format (2).


<Arithmetic instruction>

| MOVEA | Move effective address |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Effective address transfer |
| [Instruction format] | MOVEA imm16, reg1, reg2 |
| [Operation] | $GR [reg2] \leftarrow GR [reg1] + sign-extend (imm16)$ |
| [Format] | Format VI |
| [Opcode] | 15 0 31 16 rrrrr110001RRRR iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii |
| [Flags] | CY OV S Z SAT |
| [Description] | Adds the 16-bit immediate data, sign-extended to word length, to the word data of general- purpose register reg1 and stores the result in general-purpose register reg2. Neither general- purpose register reg1 nor the flags is affected. |
| [Supplement] | This instruction is to execute a 32-bit address calculation with the PSW flag value unchanged. |
| | CAUTION |
| | Do not specify r0 for reg2. |



<Arithmetic instruction>

| моуні | Move high halfword | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| | Higher halfword transfer | |
| [Instruction format] | MOVHI imm16, reg1, reg2 | |
| [Operation] | $GR [reg2] \leftarrow GR [reg1] + (imm16 \parallel 0^{16})$ | |
| [Format] | Format VI | |
| [Opcode] | 15 0 31 16 | |
| | rrrrr110010RRRR iiiiiiiiiiiiiiiiiiiiiiiiiiiii | |
| [Flags] | 07 | |
| | | |
| | s — | |
| | Z — | |
| | SAT — | |
| [Description] | Adds the word data with its higher 16 bits specified as the 16-bit immediate data and the lower 16 bits being "0" to the word data of general-purpose register reg1 and stores the result in general-purpose register reg2. Neither general-purpose register reg1 nor the flags is affected. | |
| [Supplement] | This instruction is to generate the higher 16 bits of a 32-bit address. | |
| | CAUTION | |
| | Do not specify r0 for reg2. | |



<Multiply instruction>

| <multiply instruction=""></multiply> | Multiply word by register/immediate (9-bit |
|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MUL | Multiplication of (signed) word data |
| [Instruction format] | (1) MUL reg1, reg2, reg3 |
| | (2) MUL imm9, reg2, reg3 |
| [Operation] | (1) GR [reg3] \parallel GR [reg2] \leftarrow GR [reg2] \times GR [reg1] |
| | (2) GR [reg3] \parallel GR [reg2] \leftarrow GR [reg2] \times sign-extend (imm9) |
| [Format] | (1) Format XI |
| | (2) Format XII |
| [Opcode] | |
| | 15 0.31 16 (1) rrrr111111RRRRR wwww01000100000 |
| | 15 0 31 16 (2) rrrr111111iiiii wwww01001IIII00 |
| | iiiii are the lower 5 bits of 9-bit immediate data. IIII are the higher 4 bits of 9-bit immediate data. |
| [Flags] | CY |
| | OV — |
| | s — |
| | Z — |
| | SAT — |
| [Description] | Multiplies the word data in general-purpose register reg2 by the word data in general- purpose register reg1, then stores the higher 32 bits of the result (64-bit data) in general- purpose register reg3 and the lower 32 bits in general-purpose register reg2. The contents of general-purpose registers reg1 and reg2 are handled as 32-bit signed integers. General-purpose register reg1 is not affected. |
| | (2) Multiplies the word data in general-purpose register reg2 by 9-bit immediate data, extended to word length, then stores the higher 32 bits of the result (64-bit data) in general-purpose register reg3 and the lower 32 bits in general-purpose register reg2. |
| [Supplement] | When general-purpose register reg2 and general-purpose register reg3 are the same register, |

RENESAS

only the higher 32 bits of the multiplication result are stored in the register.

RH850G3MH Software

| <multiply instruction=""></multiply> | |
|--------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| мшн | Multiply halfword by register/immediate (5-bit) |
| | Multiplication of (signed) halfword data |
| [Instruction format] | (1) MULH reg1. reg2 |
| | (2) MULH imm5, reg2 |
| [Operation] | (1) GR [reg2] \leftarrow GR [reg2] (15:0) × GR [reg1] (15:0) |
| | (2) GR [reg2] \leftarrow GR [reg2] \times sign-extend (imm5) |
| [Format] | (1) Format I |
| | (2) Format II |
| [Opcode] | |
| | 15 0 (1) rrrr000111RRRR |
| | $rrrrr \neq 00000$ (Do not specify r0 for reg2.) |
| | (2) rrrr010111iiii |
| | $rrrrr \neq 00000$ (Do not specify r0 for reg2.) |
| [Flags] | |
| [8-] | CY — |
| | ov — |
| | S — |
| | SAT — |
| | |
| [Description] | Multiplies the lower halfword data of general-purpose register reg2 by the halfword data of general-purpose register reg1 and stores the result in general-purpose register reg2. General-purpose register reg1 is not affected. |
| | (2) Multiplies the lower halfword data of general-purpose register reg2 by the 5-bit immediate data, sign-extended to halfword length, and stores the result in general- purpose register reg2. |



[Supplement]

In the case of a multiplier or a multiplicand, the higher 16 bits of general-purpose registers reg1 and reg2 are ignored.

CAUTION

Do not specify r0 for reg2.



<Multiply instruction>

| MULHI | Multiply halfword by immediate (16-bi | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| | Multiplication of (signed) halfword immediate dat | |
| [Instruction format] | MULHI imm16, reg1, reg2 | |
| [Operation] | $GR [reg2] \leftarrow GR [reg1](15:0) \times imm16$ | |
| [Format] | Format VI | |
| [Opcode] | 15 0 31 16 | |
| | rrrr110111RRRRR iiiiiiiiiiiiiiii | |
| | $rrrrr \neq 00000$ (Do not specify r0 for reg2.) | |
| [Flags] | | |
| | CY — | |
| | 0V — | |
| | 7 — | |
| | SAT — | |
| [Description] | Multiplies the lower halfword data of general-purpose register reg1 by the 16-bit immediate data and stores the result in general-purpose register reg2. General-purpose register reg1 is not affected. | |
| [Supplement] | In the case of a multiplicand, the higher 16 bits of general-purpose register reg1 are ignored. | |
| | CAUTION | |
| | Do not specify r0 for reg2. | |



RH850G3MH Software

<Multiply instruction>

| Μ | U | L | U |
|---|---|---|---|
| | | | |

Multiply word unsigned by register/immediate (9-bit)

Multiplication of (unsigned) word data

| [Instruction format] | (1) MULU reg1, reg2, reg3 |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | (2) MULU imm9, reg2, reg3 |
| [Operation] | (1) GR [reg3] \parallel GR [reg2] \leftarrow GR [reg2] \times GR [reg1] |
| | (2) GR [reg3] \parallel GR [reg2] \leftarrow GR [reg2] \times zero-extend (imm9) |
| [Format] | (1) Format XI |
| | (2) Format XII |
| [Opcode] | |
| | 15 0.31 16 (1) rrrr111111RRRR wwwww01000100010 |
| | <u>15 0 31 16</u> |
| | (2) rrrr111111iiii wwww01001IIII0 |
| | iiiii are the lower 5 bits of 9-bit immediate data. IIII are the higher 4 bits of 9-bit immediate data. |
| [Flags] | |
| | CY OV |
| | s — |
| | Z – SAT – |
| [Description] | (1) Multiplies the word data in general-purpose register reg2 by the word data in general-purpose register reg1, then stores the higher 32 bits of the result (64-bit data) in general-purpose register reg3 and the lower 32 bits in general-purpose register reg2. General-purpose register reg1 is not affected. |
| | (2) Multiplies the word data in general-purpose register reg2 by 9-bit immediate data, zero- extended to word length, then stores the higher 32 bits of the result (64-bit data) in general-purpose register reg3 and the lower 32 bits in general-purpose register reg2. |
| [Supplement] | When general-purpose register reg2 and general-purpose register reg3 are the same register, only the higher 32 bits of the multiplication result are stored in the register. |

RENESAS

<Special instruction>

| NOP | | | No operation No operation |
|----------------------|---------------------------|-------------------------------------------------|------------------------------|
| [Instruction format] | NOP | | |
| [Operation] | No operatio | on is performed. | |
| [Format] | Format I | | |
| [Opcode] | 15 0000000 | 0 0000000 | |
| [Flags] | CY OV S Z SAT | | |
| [Description] | Performs n | o processing and executes the next instruction. | |
| [Supplement] | The opcode | e is the same as that of MOV r0, r0. | |



<Logical instruction>

| ΝΟΤ | | | NOT |
|----------------------|-------------------------------------|------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|
| | | | Logical negation (1's complement) |
| [Instruction format] | NOT reg | g1, reg2 | |
| [Operation] | GR [reg2 | $[] \leftarrow NOT (GR [reg1])$ | |
| [Format] | Format I | | |
| [Opcode] | 15 rrrr0 | 0 00001RRRRR | |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwis "1" if the operation result is "0"; otherwise, "0". — | se, "0". |
| [Description] | Logically stores the affected | v negates the word data of general-purpose register result in general-purpose register reg2. General-pu | reg1 using 1's complement and rpose register reg1 is not |



<Bit manipulation instruction>

| NOT1 | | NOT bit NOT bit |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| [Instruction format] | (1) NOT1 bit#3, disp16 [reg1](2) NOT1 reg2, [reg1] | |
| [Operation] | adr ← GR [reg1] + sign-extend (disp16)*1 token ← Load-memory (adr, Byte) Z flag ← Not (extract-bit (token, bit#3)) token ← not-bit (token, bit#3) Store-memory (adr, token, Byte) | |
| | (2) adr ← GR [reg1]*1 token ← Load-memory (adr, Byte) Z flag ← Not (extract-bit (token, reg2)) token ← not-bit (token, reg2) Store-memory (adr, token, Byte) | |
| | Note 1. An MDP exception might occur depending on the result of address calculation. | |
| [Format] | (1) Format VIII (2) Format IX | |
| [Opcode] | 15 0.31 16 (1) 01bbb111110RRRRR dddddddddddddddddddddddddddddd | |
| [Flags] | CY — OV — S — Z "1" if bit specified by operand = "0", "0" if bit specified by operand = "1". SAT — | |

| [Description] | (1) Adds the word data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Byte data is read from the generated address, then the bits indicated by the 3-bit bit number are inverted (0 → 1, 1 → 0) and the data is written back to the original address. If the specified bit of the read byte data is "0", the Z flag is set to "1", and if the specifie bit is "1", the Z flag is cleared to "0". | | | |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|
| | (2) Reads the word data of general-purpose register reg1 to generate a 32-bit address. Byte data is read from the generated address, then the bits specified by lower 3 bits of general-purpose register reg2 are inverted (0 → 1, 1 → 0) and the data is written back to the original address. If the specified bit of the read byte data is "0", the Z flag is set to "1", and if the specified bit is "1", the Z flag is cleared to "0". | | | |
| [Supplement] | The Z flag of PSW indicates the status of the specified bit (0 or 1) before this instruction is executed and does not indicate the content of the specified bit resulting from the instruction execution. | | | |

CAUTION

This instruction provides an atomic guarantee aimed at exclusive control, and during the period between read and write operations, the target address is not affected by access due to any other cause.



<Logical instruction>

| OR | | OR |
|----------------------|----------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | OR reg1, | reg2 |
| [Operation] | GR [reg2 | $] \leftarrow GR [reg2] OR GR [reg1]$ |
| [Format] | Format I | |
| [Opcode] | 15 rrrrr0 | 0 01000RRRR |
| [Flags] | CY OV S Z SAT | 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". |
| [Description] | ORs the v register re reg1 is no | word data of general-purpose register reg2 with the word data of general-purpose eg1 and stores the result in general-purpose register reg2. General-purpose register t affected. |



<Logical instruction>

| ORI | | OR in | nmediate (16-bit) |
|----------------------|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| | | | OR immediate |
| [Instruction format] | ORI imm | n16, reg1, reg2 | |
| [Operation] | GR [reg2] |] \leftarrow GR [reg1] OR zero-extend (imm16) | |
| [Format] | Format V | Ί | |
| [Opcode] | 15 rrrrl | 0 31 16 10100RRRRR iiiiiiiiiiiiiiii | |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". — | |
| [Description] | ORs the v extended purpose re | word data of general-purpose register reg1 with the 16-bit immediate to word length, and stores the result in general-purpose register reg2. egister reg1 is not affected. | data, zero- General- |



<Special instruction>

Pop registers from Stack POPSP POP from the stack [Instruction format] POPSP rh-rt [Operation] if $rh \leq rt$ then $cur \leftarrow rt$ end \leftarrow rh $tmp \leftarrow sp$ while $(cur \ge end)$ { adr $\leftarrow \operatorname{tmp}^{*1, *2}$ $GR[cur] \leftarrow Load-memory (adr, Word)$ $cur \leftarrow cur - 1$ $tmp \leftarrow tmp + 4$ } sp ← tmp Note 1. An MDP exception might occur depending on the result of address calculation. Note 2. The lower 2 bits of adr are masked to 0. [Format] Format XI [Opcode] 15 0 31 16 01100111111RRRRR wwwww00101100000 RRRRR indicates rh. wwww indicates rt. [Flags] CY ov s Ζ SAT [Description] Loads general-purpose register rt to rh from the stack in descending order (rt, rt -1, rt -2, ..., rh). After all the registers down to the specified register have been loaded, sp is updated

[Supplement]

The lower two bits of the address specified by sp are masked by 0. If an exception is acknowledged before sp is updated, instruction execution is halted and exception handling is executed with the start address of this instruction used as the return address. The POPSP instruction is then executed again. (The sp value from before the exception handling is saved.)

CAUTION

If a register that includes sp(r3) is specified as the restore register (rh = 3 to 31), the value read from the memory is not stored in sp(r3). This allows the POPSP instruction to be correctly reexecuted after execution has been halted.



<Special instruction>

| PREPARE | | | Function prepare |
|----------------------|-----|------------------------------------------------------------------------------|--------------------|
| | | | Create stack frame |
| [Instruction format] | (1) | PREPARE list12, imm5 | |
| | (2) | PREPARE list12, imm5, sp/imm ' | |
| | | Note 1. The sp/imm values are specified by bits 19 and 20 of the sub-opcode. | |
| [Operation] | (1) | $tmp \leftarrow sp$ | |
| | | foreach (all regs in list12) { | |
| | | $tmp \leftarrow tmp - 4$ | |
| | | $adr \leftarrow tmp^{-1}, 2$ | |
| | | Store-memory (adr, GR[reg in list12], Word) | |
| | | } | |
| | | $sp \leftarrow tmp - zero-extend (imm5 logically shift left by 2)$ | |
| | (2) | $tmp \leftarrow sp$ | |
| | | foreach (all regs in list12) { | |
| | | $tmp \leftarrow tmp - 4$ | |
| | | Store-memory (adr. GR[reg in list12], Word) | |
| | | } | |
| | | $sp \leftarrow tmp - zero-extend (imm5 logically shift left by 2)$ | |
| | | case | |
| | | $ff = 00: ep \leftarrow sp$ | |
| | | $ff = 01: ep \leftarrow sign-extend (imm16)$ | |
| | | ff = 10: ep \leftarrow imm16 logically shift left by 16 | |
| | | $ff = 11: ep \leftarrow imm32$ | |
| | | Note 1. An MDP exception might occur depending on the result of address ca | lculation. |
| | | Note 2. The lower 2 bits of adr are masked to 0. | |
| [Format] | For | mat XIII | |



[Opcode]

| | 15 0 |) 31 | 16 |
|-----|-----------------|---------------|----------------------------|
| (1) | 0000011110iiiiI | LLLLLLLLL0000 | 01 |
| | _15(|) 31 | 16 Option (47-32 or 63-32) |
| (2) | 0000011110iiiiI | LLLLLLLLLff01 | 1 imm16/imm32 |

In the case of 32-bit immediate data (imm32), bits 47 to 32 are the lower 16 bits of imm32 and bits 63 to 48 are the higher 16 bits of imm32.

ff = 00: sp is loaded to ep

ff = 01: Sign-extended 16-bit immediate data (bits 47 to 32) is loaded to ep

ff = 10: 16-bit logical left-shifted 16-bit immediate data (bits 47 to 32) is loaded to ep

ff = 11: 32-bit immediate data (bits 63 to 32) is loaded to ep

The values of LLLLLLLLLL are the corresponding bit values shown in register list "list12" (for example, the "L" at bit 21 of the opcode corresponds to the value of bit 21 in list12).

list12 is a 32-bit register list, defined as follows.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|-----|
| r24 | r25 | r26 | r27 | r20 | r21 | r22 | r23 | r28 | r29 | r31 | _ | r30 |

Bits 31 to 21 and bit 0 correspond to general-purpose registers (r20 to r31), so that when any of these bits is set (1), it specifies a corresponding register operation as a processing target. For example, when r20 and r30 are specified, the values in list12 appear as shown below (register bits that do not correspond, i.e., bits 20 to 1 are set as "Don't care").

- When all of the register's non-corresponding bits are "0": $0800\ 0001_{\rm H}$
- When all of the register's non-corresponding bits are "1": $081F FFF_H$

[Flags]

| CY | _ |
|-----|---|
| OV | — |
| S | — |
| Z | — |
| SAT | — |

| [Description] | (1) Saves general-purpose registers specified in list12 (4 is subtracted from the sp value and the data is stored in that address). Next, subtracts 5-bit immediate data, logically left- shifted by 2 bits and zero-extended to word length, from sp. |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | (2) Saves general-purpose registers specified in list12 (4 is subtracted from the sp value and the data is stored in that address). Next, subtracts 5-bit immediate data, logically left-shifted by 2 bits and zero-extended to word length, from sp. Then, loads the data specified by the third operand (sp/imm) to ep. |
| [Supplement] | list12 general-purpose registers are saved in ascending order (r20, r21,, r31). imm5 is used to create a stack frame that is used for auto variables and temporary data. The lower two bits of the address specified by sp are masked to 0 and aligned to the word boundary. |
| | CAUTION |

If an exception occurs while this instruction is being executed, execution of the instruction might be stopped after the write cycle and the register value write operation are completed, but sp will retain its original value from before the start of execution. The instruction will be executed again later, after a return from the exception.



<Special instruction>

PUSHSP

Push registers to Stack

Push registers to Stack

| [Instruction format] | PUSHSP rh-rt | | | | | | |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|--|--|--|
| [Operation] | if $rh \leq rt$ | | | | | | |
| | then $\operatorname{cur} \leftarrow \operatorname{rh}$ | | | | | | |
| | end \leftarrow rt | | | | | | |
| | tmp ← sp | | | | | | |
| | while ($cur \leq end$) { | | | | | | |
| | $tmp \leftarrow tmp - 4$ | | | | | | |
| | $adr \leftarrow tmp^{*1,*2}$ | | | | | | |
| | Store-memory (adr, GR[cur], Word) | | | | | | |
| | $cur \leftarrow cur + 1$ | | | | | | |
| | } | | | | | | |
| | $sp \leftarrow tmp$ | | | | | | |
| | Note 1. An MDP exception might occur depending on the result of address calculation.Note 2. The lower 2 bits of adr are masked to 0. | | | | | | |
| [Format] | Format XI | | | | | | |
| [Opcode] | 15 0 31 16 01000111111RRRRR wwww00101100000 | | | | | | |
| | RRRR indicates rh. wwww indicates rt. | | | | | | |
| [Flags] | | | | | | | |
| | CY — | | | | | | |
| | S — | | | | | | |
| | Z — | | | | | | |
| | SAT — | | | | | | |
| [Description] | Stores general-purpose register rh to rt in the stack in ascending order (rh, rh +1, rh + 2,, rt). After all the specified registers have been stored, sp is updated (decremented). | | | | | | |

RENESAS

[Supplement]

The lower two bits of the address specified by sp are masked by 0. If an exception is acknowledged before sp is updated, instruction execution is halted and exception handling is executed with the start address of this instruction used as the return address. The PUSHSP instruction is then executed again. (The sp value from before the exception handling is saved.)



Special instruction

| <special instruction=""></special> | Reserved instruction exception |
|------------------------------------|-----------------------------------------------------------------------------------------------------------|
| RIE | |
| | Reserved instruction exception |
| [Instruction format] | (1) BIE |
| | (2) RIE imm5, imm4 |
| | |
| [Operation] | $FEPC \leftarrow PC \text{ (return PC)}$ |
| | $FEPSW \leftarrow PSW$ |
| | FEIC \leftarrow exception cause code (0000 0060 _H) PSW LIM $\leftarrow 0$ |
| | $PSW.NP \leftarrow 1$ |
| | $PSW.EP \leftarrow 1$ |
| | $\text{PSW.ID} \leftarrow 1$ |
| | PC \leftarrow exception handler address (offset address 60_{H}) |
| [Format] | (1) Format I |
| [1 011110] | (2) Format X |
| | |
| [Opcode] | 15 0 |
| | |
| | |
| | 15 031 16 |
| | (2) $i i i i i 1 1 1 1 1 1 1 1 1 1 1 1 1 1 $ |
| | (2) |
| | Where iiiii = imm5, IIII = imm4. |
| (15) | |
| [Flags] | CY — |
| | ov — |
| | S — |
| | Z — |
| | SAT — |
| [Description] | Saves the contents of the return PC (address of the RIE instruction) and the current contents of |
| r sametrand | the PSW to FEPC and FEPSW, respectively, stores the exception cause code in the FEIC |
| | register, and updates the PSW according to the exception causes listed in Table 4.1 . |
| | Execution then branches to the exception handler address and exception handling is started. |
| | Exception handler addresses are calculated based on the offset address 60 _H . For details, see |

Section 4.4, Exception Handler Address.

<Data manipulation instruction>

| ROTL | Rotate Left |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | (1) ROTL imm5, reg2, reg3 |
| | (2) ROTL reg1, reg2, reg3 |
| [Operation] | (1) $GR[reg3] \leftarrow GR[reg2]$ rotate left by zero-extend (imm5) |
| | (2) $GR[reg3] \leftarrow GR[reg2]$ rotate left by $GR[reg1]$ |
| [Format] | Format VII |
| [Opcode] | 15 0 31 16 |
| | (1) rrrr111111iiii wwww00011000100 |
| | 15 0 31 16 (2) rrrr111111RRRRR wwwww00011000110 |
| [Flags] | |
| | CY "1" if operation result bit 0 is "1"; otherwise "0", including if the rotate amount is "0". |
| | OV U |
| | Z "1" if the operation result is "0": otherwise, "0". |
| | SAT — |
| [Description] | (1) Rotates the word data of general-purpose register reg2 to the left by the specified shift amount, which is indicated by a 5-bit immediate value zero-extended to word length. The result is written to general-purpose register reg3. General-purpose register reg2 is not affected. |
| | (2) Rotates the word data of general-purpose register reg2 to the left by the specified shift amount indicated by the lower 5 bits of general-purpose register reg1. The result is written to general-purpose register reg3. General-purpose registers reg1 and reg2 are not affected. |



<Data manipulation instruction>

| SAR | Shift arithmetic right by register/immediate (5-bit) |
|----------------------|------------------------------------------------------------------------------------------|
| | Arithmetic right shift |
| [Instruction format] | (1) SAR reg1, reg2 |
| | (2) SAR imm5, reg2 |
| | (3) SAR reg1, reg2, reg3 |
| [Operation] | (1) GR [reg2] \leftarrow GR [reg2] arithmetically shift right by GR [reg1] |
| | (2) GR [reg2] \leftarrow GR [reg2] arithmetically shift right by zero-extend (imm5) |
| | (3) GR [reg3] \leftarrow GR [reg2] arithmetically shift right by GR [reg1] |
| [Format] | (1) Format IX |
| | (2) Format II |
| | (3) Format XI |
| [Opcode] | |
| - | 15 0.31 16 (1) rrrr111111RRRRR 000000010100000 |
| | 15 0 (2) rrrr010101iiii |
| | 15 0 31 16 (3) rrrr111111RRRRR wwwww00010100010 |
| [Flags] | |
| [* ***60] | CY "1" if the last bit shifted out is "1"; otherwise, "0" including non-shift. |
| | OV 0 |
| | S "1" if the operation result is negative; otherwise, "0". |
| | SAT — |



[Description]

- Arithmetically right-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the lower 5 bits of general-purpose register reg1, by copying the pre-shift MSB value to the post-shift MSB. The result is written to general-purpose register reg2. General-purpose register reg1 is not affected.
- (2) Arithmetically right-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the 5-bit immediate data, zero-extended to word length, by copying the pre-shift MSB value to the post-shift MSB. The result is written to general-purpose register reg2.
- (3) Arithmetically right-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the lower 5 bits of general-purpose register reg1, by copying the pre-shift MSB value to the post-shift MSB. The result is written to general-purpose register reg3. General-purpose registers reg1 and reg2 are not affected.



<Data manipulation instruction>

| SASF | | | | | | Shift and set flag conditior |
|----------------------|----------------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|-------------------------------------------------------------------|----------------------------------------------------------------------|
| | | | | | S | hift and flag condition setting |
| [Instruction format] | SASF | cccc, reg2 | | | | |
| [Operation] | if cond then (else (| itions are sat GR [reg2] ← GR [reg2] ← | isfied (GR [reg2] Logically sl (GR [reg2] Logically sh | hift left by 1) hift left by 1) |) OR 0000 0 OR 0000 00 | 001 _H 000 _H |
| [Format] | Format | t IX | | | | |
| [Opcode] | 15 rrrri | r1111110c | 031 :ccc 000000100000 | 16 0000 | | |
| [Flags] | CY OV S Z SAT | | | | | |
| [Description] | When t general conditi Design | the condition l-purpose reg on is not me ate one of th | specified by condition gister reg2 by 1 bit, and s t, logically left-shifts da e condition codes shown | code "cccc" sets (1) the le ta of reg2 an n in the follo | is met, logic east significa d clears the l wing table a | ally left-shifts data of ant bit (LSB). If a LSB. s [cccc]. |
| | Condition Code | n Name | Condition Formula | Condition Code | Name | Condition Formula |
| | 0000 | V | OV = 1 | 0100 | S/N | S = 1 |
| | 1000 | NV | OV = 0 | 1100 | NS/P | S = 0 |
| | 0001 | C/L | CY = 1 | 0101 | Т | Always (unconditional) |
| | 1001 | NC/NL | CY = 0 | 1101 | SA | SAT = 1 |
| | 0010 | Z | Z = 1 | 0110 | LT | (S xor OV) = 1 |
| | 1010 | NZ | Z = 0 | 1110 | GE | (S xor OV) = 0 |

[Supplement]

See the SETF instruction.

NH

н

0011

1011

0111

1111

LE

GT

(CY or Z) = 1

(CY or Z) = 0

((S xor OV) or Z) = 1

((S xor OV) or Z) = 0

<Saturated operation instructions>

| SATADD | | Saturated add register | r/immediate (5-bit) |
|----------------------|---------|----------------------------------------------------------------|---------------------|
| | | | Saturated addition |
| [Instruction format] | (1) SAT | TADD reg1, reg2 | |
| | (2) SAT | TADD imm5, reg2 | |
| | (3) SAT | TADD reg1, reg2, reg3 | |
| [Operation] | (1) GR | $[reg2] \leftarrow saturated (GR [reg2] + GR [reg1])$ | |
| | (2) GR | $[reg2] \leftarrow saturated (GR [reg2] + sign-extend (imm5))$ | |
| | (3) GR | $[reg3] \leftarrow saturated (GR [reg2] + GR [reg1])$ | |
| [Format] | (1) For | mat I | |
| | (2) For | mat II | |
| | (3) For | rmat XI | |
| [Opcode] | 15 | 5 O | |
| | (1) ri | rrrr000110RRRR | |
| | rrrrr 7 | \neq 00000 (Do not specify r0 for reg2.) | |
| | 15 | 5 0 | |
| | (2) ri | rrrr01000liiiii | |
| | rrrrr ; | \neq 00000 (Do not specify r0 for reg2.) | |
| | 15 | 5 031 16 | |
| | (3) ri | rrrr111111RRRRR wwww01110111010 | |
| [Flags] | | | |
| [* ***60] | CY | "1" if a carry occurs from MSB; otherwise, "0". | |
| | OV | "1" if overflow occurs; otherwise, "0". | |
| | S | "1" if saturated operation result is negative; otherwise, "0". | |
| | Z | "1" if saturated operation result is "0"; otherwise, "0". | |
| | SAT | "1" if OV = 1; otherwise, does not change. | |



| [Description] | (1) Adds the word data of general-purpose register reg1 to the word data of general-purpose register reg2, and stores the result in general-purpose register reg2. However, when the result exceeds the maximum positive value 7FFF FFFF _H , 7FFF FFFF _H is stored in reg2, and when it exceeds the maximum negative value 8000 0000 _H , 8000 0000 _H is stored in reg2; then the SAT flag is set (1). General-purpose register reg1 is not affected. |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | (2) Adds the 5-bit immediate data, sign-extended to the word length, to the word data of general-purpose register reg2, and stores the result in general-purpose register reg2. However, when the result exceeds the maximum positive value 7FFF FFFF_H, 7FFF FFFF_H is stored in reg2, and when it exceeds the maximum negative value 8000 0000_H, 8000 0000_H is stored in reg2; then the SAT flag is set (1). |
| | (3) Adds the word data of general-purpose register reg1 to the word data of general-purpose register reg2, and stores the result in general-purpose register reg3. However, when the result exceeds the maximum positive value 7FFF FFFF _H , 7FFF FFFF _H is stored in reg3, and when it exceeds the maximum negative value 8000 0000 _H , 8000 0000 _H is stored in reg3; then the SAT flag is set (1). General-purpose registers reg1 and reg2 are not affected. |
| [Supplement] | The SAT flag is a cumulative flag. The saturate result sets the flag to "1" and will not be cleared to "0" even if the result of the subsequent operation is not saturated. The saturated operation instruction is executed normally, even with the SAT flag set to "1". |
| | CAUTIONS |

- 1. Use LDSR instruction and load data to the PSW to clear the SAT flag to "0".
- 2. Do not specify r0 as reg2 in instruction format (1) SATADD reg1, reg2 and in instruction format (2) SATADD imm5, reg2.



<Saturated operation instructions>

| GATCHB | Saturated subtract |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SAISUB | Saturated subtraction |
| [Instruction format] | (1) SATSUB reg1, reg2 |
| | (2) SATSUB reg1, reg2, reg3 |
| [Operation] | (1) GR [reg2] \leftarrow saturated (GR [reg2] – GR [reg1]) |
| | (2) GR [reg3] \leftarrow saturated (GR [reg2] – GR [reg1]) |
| [Format] | (1) Format I |
| | (2) Format XI |
| [Opcode] | 15 0 |
| | (1) rrrr000101RRRR |
| | $rrrrr \neq 00000$ (Do not specify r0 for reg2.) |
| | 15 031 16 |
| | (2) rrrriillinkkkk www.wolliooliolo |
| [Flags] | |
| | CY "1" if a borrow occurs from MSB; otherwise, "0". |
| | OV "1" if overflow occurs; otherwise, "0". |
| | S "1" if saturated operation result is negative; otherwise, "0". |
| | Z "1" if saturated operation result is "0"; otherwise, "0". |
| | SAT "1" if OV = 1; otherwise, does not change. |
| [Description] | Subtracts the word data of general-purpose register reg1 from the word data of general-purpose register reg2 and stores the result in general-purpose register reg2. If the result exceeds the maximum positive value 7FFF FFFF_H, 7FFF FFFF_H is stored in reg2; if the result exceeds the maximum negative value 8000 0000_H, 8000 0000_H is stored in reg2. The SAT flag is set to "1". General-purpose register reg1 is not affected. |
| | (2) Subtracts the word data of general-purpose register reg1 from the word data of general-purpose register reg2, and stores the result in general-purpose register reg3. However, when the result exceeds the maximum positive value 7FFF FFFF_H, 7FFF FFFF_H is stored in reg3, and when it exceeds the maximum negative value 8000 0000_H, 8000 0000_H is stored in reg3; then the SAT flag is set (1). General-purpose registers reg1 and reg2 are not affected. |

RENESAS

[Supplement] The SAT flag is a cumulative flag. The saturate result sets the flag to "1" and will not be cleared to "0" even if the result of the subsequent operation is not saturated. The saturated operation instruction is executed normally, even with the SAT flag set to "1".

CAUTIONS

- 1. Use LDSR instruction and load data to the PSW to clear the SAT flag to "0".
- 2. Do not specify r0 as reg2 in instruction format (1) SATSUB reg1, reg2.



<Saturated operation instructions>

| SATSUBI | Saturated subtract immediate | | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| [Instruction format] | SATSUBI imm16, reg1, reg2 | | |
| [Operation] | GR [reg2] \leftarrow saturated (GR [reg1] – sign-extend (imm16)) | | |
| [Format] | Format VI | | |
| [Opcode] | 15 031 16 rrrr110011RRRRR iiiiiiiiiiiiii | | |
| | $rrrrr \neq 00000$ (Do not specify r0 for reg2.) | | |
| [Flags] | CY "1" if a borrow occurs from MSB; otherwise, "0". OV "1" if overflow occurs; otherwise, "0". S "1" if saturated operation result is negative; otherwise, "0". Z "1" if saturated operation result is "0"; otherwise, "0". SAT "1" if OV = 1; otherwise, does not change. | | |
| [Description] | Subtracts the 16-bit immediate data, sign-extended to word length, from the word data of general-purpose register reg1 and stores the result in general-purpose register reg2. If the result exceeds the maximum positive value 7FFF $FFFF_H$, 7FFF $FFFF_H$ is stored in reg2; if the result exceeds the maximum negative value 8000 0000 _H , 8000 0000 _H is stored in reg2. The SAT flag is set to "1". General-purpose register reg1 is not affected. | | |
| [Supplement] | The SAT flag is a cumulative flag. The saturation result sets the flag to "1" and will not be cleared to "0" even if the result of the subsequent operation is not saturated. The saturated operation instruction is executed normally, even with the SAT flag set to "1". | | |
| | CAUTIONS | | |
| | Use LDSR instruction and load data to the PSW to clear the SAT flag to "0". Do not specify r0 for reg2. | | |



<Saturated operation instructions>

| CATCHER | | Saturated subtract reverse | |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| SAISUBI | | Saturated reverse subtraction | |
| [Instruction format] | SATSUBR 1 | reg1, reg2 | |
| [Operation] | GR [reg2] ← | - saturated (GR [reg1] – GR [reg2]) | |
| [Format] | Format I | | |
| [Opcode] | 15 rrrrr000 rrrrr≠00 | 0 100RRRR 000 (Do not specify r0 for reg2.) | |
| [Flags] | CY OV S Z SAT | "1" if a borrow occurs from MSB; otherwise, "0". "1" if overflow occurs; otherwise, "0". "1" if saturated operation result is negative; otherwise, "0". "1" if saturated operation result is "0"; otherwise, "0". "1" if OV = 1; otherwise, does not change. | |
| [Description] | Subtracts the word data of general-purpose register reg2 from the word data of general- purpose register reg1 and stores the result in general-purpose register reg2. If the result exceeds the maximum positive value 7FFF FFFF _H , 7FFF FFFF _H is stored in reg2; if the result exceeds the maximum negative value 8000 0000 _H , 8000 0000 _H is stored in reg2. The SAT flag is set to "1". General-purpose register reg1 is not affected. | | |
| [Supplement] | The SAT flag is a cumulative flag. The saturation result sets the flag to "1" and will not be cleared to "0" even if the result of the subsequent operation is not saturated. The saturated operation instruction is executed normally, even with the SAT flag set to "1". | | |
| | CAUTIONS | | |
| | 1. Use LD 2. Do not | DSR instruction and load data to the PSW to clear the SAT flag to "0". specify r0 for reg2. | |



٦

<Conditional operation instructions>

| SBF | | | | | | Subtract on condition flag |
|----------------------|-------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | | | | | Conditional subtraction |
| [Instruction format] | SBF ccc | c, reg1, reg | g2, reg3 | | | |
| [Operation] | if condit then GR else GR | ions are sat [reg3] ← ([reg3] ← (| iisfied GR [reg2] – GR [reg1] – GR [reg2] – GR [reg1] –(| -1 0 | | |
| [Format] | Format 2 | ΚI | | | | |
| [Opcode] | 15 rrrri | 111111RF | 031 RRR wwww011100c | 16 ccc0 | | |
| [Flags] | CY OV S Z SAT | "1" if "1" if "1" if — | a borrow occurs from MSB overflow occurs; otherwise, operation result is negative operation result is "0"; othe | ; otherwise, "0". , "0". ; otherwise, "0". rwise, "0". | | |
| [Description] | Subtract the word purpose If the co- general-j the resul General- Designat cannot e | s 1 from the data of ge register reg ndition spe purpose reg t in general purpose re purpose re purpose re te one of th qual 1101.) | e result of subtracting the neral-purpose register re g3, if the condition speci cified by condition code gister reg1 from the word l-purpose register reg3. gisters reg1 and register e condition codes shown | e word data of g2, and stores fied by condit "cccc" is not d data of gener 2 are not affect in the follow | f general-put the result ion code "c satisfied, s ral-purpose eted. ing table as | arpose register reg1 from of subtraction in general- eccc" is satisfied. ubtracts the word data of register reg2, and stores s [cccc]. (However, cccc |
| | Condition Code | Name | Condition Formula | Condition Code | Name | Condition Formula |
| | 0000 | V | OV = 1 | 0100 | S/N | S = 1 |
| | 1000 | NV | OV = 0 | 1100 | NS/P | S = 0 |
| | 0001 | C/L | CY = 1 | 0101 | Т | Always (Unconditional) |
| | 1001 | NC/NL | CY = 0 | 0110 | LT | (S xor OV) = 1 |
| | 0010 | Z | Z = 1 | 1110 | GE | (S xor OV) = 0 |
| | 1010 | NZ | Z = 0 | 0111 | LE | ((S xor OV) or Z) = 1 |
| | 0011 | NH | (CY or Z) = 1 | 1111 | GT | ((S xor OV) or Z) = 0 |
| | 1011 | Н | (CY or Z) = 0 | (1101) | Setting pr | ohibited |



| SCHOL | Search zero from | | |
|----------------------|------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|
| | | | Bit (0) search from MSB side |
| [Instruction format] | SCH0L re | eg2, reg3 | |
| [Operation] | GR [reg3] | ← search zero from left of GR [reg2] | |
| [Format] | Format IX | | |
| [Opcode] | 15 rrrrr1 | 031 16 | |
| [Flags] | CY OV S Z SAT | "1" if bit (0) is found eventually; otherwise, "0". 0 "1" if bit (0) is not found; otherwise, "0". — | |
| [Description] | Searches w the numbe purpose re When bit bit (0) fou | word data of general-purpose register reg2 from the left er of 1s before the bit position (0 to 31) at which 0 is fur- egister reg3 (e.g., when bit 31 of reg2 is 0, $01_{\rm H}$ is writte (0) is not found, 0 is written to reg3, and the Z flag is s nd is the LSB, the CY flag is set (1). | side (MSB side), and writes rst found plus 1 to general- en to reg3). imultaneously set (1). If the |



| SCH0R | | | Search zero from right |
|----------------------|------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| | | | Bit (0) search from LSB side |
| [Instruction format] | SCH0R r | eg2, reg3 | |
| [Operation] | GR [reg3] | ← search zero from right of GR [reg2] | |
| [Format] | Format IX | | |
| [Opcode] | 15 rrrr1 | 031 16 1111100000 wwww01101100000 | |
| [Flags] | CY OV S Z SAT | "1" if bit (0) is found eventually; otherwise, "0". 0 "1" if bit (0) is not found; otherwise, "0". — | |
| [Description] | Searches w the numbe purpose re When bit bit (0) fou | word data of general-purpose register reg2 from the right e^{2} er of 1s before the bit position (0 to 31) at which 0 is first egister reg3 (e.g., when bit 0 of reg2 is 0, 01 _H is written t (0) is not found, 0 is written to reg3, and the Z flag is sin nd is the MSB, the CY flag is set (1). | side (LSB side), and writes t found plus 1 to general- to reg3). nultaneously set (1). If the |



| SCH1L | | Search one from left |
|----------------------|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Bit (1) search from MSB side |
| [Instruction format] | SCH1L r | eg2, reg3 |
| [Operation] | GR [reg3] | $] \leftarrow$ search one from left of GR [reg2] |
| [Format] | Format IX | X |
| [Opcode] | 15 | 0 31 16 |
| | rrrrr1 | 1111100000 wwww01101100110 |
| [Flags] | | |
| | CY | "1" if bit (0) is found eventually; otherwise, "0". |
| | S | 0 |
| | z | "1" if bit (0) is not found: otherwise. "0". |
| | SAT | _ |
| [Description] | Searches the numb purpose r When bit bit (1) fou | word data of general-purpose register reg2 from the left side (MSB side), and writes er of 0s before the bit position (0 to 31) at which 1 is first found plus 1 to general- egister reg3 (e.g., when bit 31 of reg2 is 1, $01_{\rm H}$ is written to reg3). (1) is not found, 0 is written to reg3, and the Z flag is simultaneously set (1). If the and is the LSB, the CY flag is set (1). |



| SCH1R | | Search one from right |
|----------------------|---------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Bit (1) search from LSB side |
| [Instruction format] | SCH1R re | eg2, reg3 |
| [Operation] | GR [reg3] | \leftarrow search one from right of GR [reg2] |
| [Format] | Format IX | |
| [Opcode] | 15 | 0 31 16 |
| | rrrrrll | 111100000 wwww01101100010 |
| [Flags] | СҮ | "1" if bit (0) is found eventually; otherwise, "0". |
| | OV | 0 |
| | S | 0 |
| | Z | "1" if bit (0) is not found; otherwise, "0". |
| | SAT | _ |
| [Description] | Searches w the numbe purpose re When bit (bit (1) four | word data of general-purpose register reg2 from the right side (LSB side), and writes r of 0s before the bit position (0 to 31) at which 1 is first found plus 1 to general- gister reg3 (e.g., when bit 0 of reg2 is 1, $01_{\rm H}$ is written to reg3). (1) is not found, 0 is written to reg3, and the Z flag is simultaneously set (1). If the nd is the MSB, the CY flag is set (1). |


<Bit manipulation instruction>

| SET1 | | | Set bit |
|----------------------|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| | | | Bit setting |
| [Instruction format] | (1) (2) | SET1 bit#3, disp16 [reg1] | |
| | (2) | 3E11 leg2, [leg1] | |
| [Operation] | (1) | adr ← GR [reg1] + sign-extend (disp16) ^{*1} token ← Load-memory (adr, Byte) Z flag ← Not (extract-bit (token, bit#3)) token ← set-bit (token, bit#3) Store-memory (adr, token, Byte) | |
| | (2) | adr \leftarrow GR [reg1] ^{*1} token \leftarrow Load-memory (adr, Byte) Z flag \leftarrow Not (extract-bit (token, reg2)) token \leftarrow set-bit (token, reg2) Store-memory (adr, token, Byte) | |
| | | Note 1. An MDP exception might occur depending on the result of address calculation. | |
| [Format] | (1) (2) | Format VIII Format IX | |
| [Opcode] | | 15 0 31 16 | |
| | (1) | 00bbb111110RRRRR ddddddddddddddd | |
| | (2) | 15 031 16 rrrr111111RRRR 000000011100000 | |
| [Flags] | CY OV S Z SA | — — — "1" if bit specified by operand = "0", "0" if bit specified by operand = "1". T — | |

| [Description] | (1) Adds the word data of general-purpose register reg1 to the16-bit displacement data, sign-extended to word length, to generate a 32-bit address. Byte data is read from the generated address, the bits indicated by the 3-bit bit number are set (1) and the data is written back to the original address. If the specified bit of the read byte data is "0", the Z flag is set to "1", and if the specified bit is "1", the Z flag is cleared to "0". |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | (2) Reads the word data of general-purpose register reg1 to generate a 32-bit address. Byte data is read from the generated address, the lower 3 bits indicated of general-purpose register reg2 are set (1) and the data is written back to the original address. If the specified bit of the read byte data is "0", the Z flag is set to "1", and if the specified bit is "1", the Z flag is cleared to "0". |
| [Supplement] | The Z flag of PSW indicates the initial status of the specified bit (0 or 1) and does not indicate the content of the specified bit resulting from the instruction execution. |
| | CAUTION |
| | This instruction provides an atomic guarantee aimed at exclusive control, and during the period |

other cause.

between read and write operations, the target address is not affected by access due to any



<Data manipulation instruction>

| SETF | | | | | | | Set flag condition |
|----------------------|---------------------------------------------|-------------------------------------|------------------------------------------------------------|--------------------------------------------|----------------------------------------------|-------------------------------------------------|-----------------------------------------------|
| | | | | | | | Flag condition setting |
| [Instruction format] | SETF cccc, | , reg2 | | | | | |
| [Operation] | if conditions then GR [r else GR [re | s are sati eg2] ← eg2] ← (| isfied 0000 0001 _H 0000 0000 _H | | | | |
| [Format] | Format IX | | | | | | |
| [Opcode] | 15 rrrrr111 | .1110c | 031 ccc 000000 | 00000000 | 16)000 | | |
| [Flags] | CY OV S Z SAT | | | | | | |
| [Description] | When the co register reg2 Designate o | ondition 2 if a cor ne of the | specified by condition is met a | ondition co and stores " les shown i | de "cccc" is 0" if a cond n the follow | s met, stores lition is not ving table as | s "1" to general-purpose met. s [cccc]. |
| | Condition Code | lame | Condition F | ormula | Condition Code | Name | Condition Formula |

| Condition | Name | Condition Formula | Condition | Name | Condition Formula |
|-----------|-------|-------------------|-----------|------|--------------------------|
| 0000 | V | OV = 1 | 0100 | S/N | S = 1 |
| 1000 | NV | OV = 0 | 1100 | NS/P | S = 0 |
| 0001 | C/L | CY = 1 | 0101 | Т | Always (Unconditional) |
| 1001 | NC/NL | CY = 0 | 1101 | SA | SAT = 1 |
| 0010 | Z | Z = 1 | 0110 | LT | (S xor OV) = 1 |
| 1010 | NZ | Z = 0 | 1110 | GE | (S xor OV) = 0 |
| 0011 | NH | (CY or Z) = 1 | 0111 | LE | ((S xor OV) or Z) = 1 |
| 1011 | Н | (CY or Z) = 0 | 1111 | GT | ((S xor OV) or Z) = 0 |



[Supplement]

Examples of SETF instruction:

(1) Translation of multiple condition clauses

If A of statement if(A) in C language consists of two or greater condition clauses (a₁, a₂, a₃, and so on), it is usually translated to a sequence of $if(a_1)$ then, $if(a_2)$ then. The object code executes "conditional branch" by checking the result of evaluation equivalent to a_n. Because a pipeline operation requires more time to execute "condition judgment" + "branch" than to execute an ordinary operation, the result of evaluating each condition clause $if(a_n)$ is stored in register Ra. By performing a logical operation to Ra_n after all the condition clauses have been evaluated, the pipeline delay can be prevented.

(2) Double-length operation

To execute a double-length operation, such as "Add with Carry", the result of the CY flag can be stored in general-purpose register reg2. Therefore, a carry from the lower bits can be represented as a numeric value.



<Data manipulation instruction>

| | | | | Shift lo | gical left by registe | er/immediate (5-bit) |
|----------------------|----------|-----------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------|----------------------|
| SHL | | | | | | Logical loft shift |
| | | | | | | Logical left shift |
| [Instruction format] | (1) SHL | reg1, reg2 | | | | |
| | (2) SHL | imm5, reg2 | | | | |
| | (3) SHL | reg1, reg2, reg | 3 | | | |
| [Operation] | (1) GR [| reg2] ← GR [re | eg2] logically sł | nift left by GR [reg1] |] | |
| | (2) GR [| reg2] ← GR [re | eg2] logically sh | nift left by zero-exter | nd (imm5) | |
| | (3) GR [| reg3] ← GR [re | eg2] logically sł | nift left by GR [reg1 |] | |
| [Format] | (1) Form | nat IX | | | | |
| | (2) Form | nat II | | | | |
| | (3) Form | nat XI | | | | |
| | | | | | | |
| [Opcode] | | | | | | |
| | (1) 75 | rrr1111111 | | 16 | | |
| | (1) | | KKK 000000 | 0011000000 | | |
| | 15 | | 0 | | | |
| | (2) rr | rrr010110i: | iiii | | | |
| | 15 | | 0 31 | 16 | | |
| | (3) rr | rrr111111R | RRRR wwww0 | 0011000010 | | |
| | <u> </u> | | | | | |
| [Flags] | | | | | | |
| | CY | "1" if the las | at bit shifted out is | "1"; otherwise, "0" inclu | uding non-shift. | |
| | ov | 0 | and the second state | and the set of the set | | |
| | 5 | "1" if the op | eration result is n | egative; otnerwise, "0". | | |
| | SAT | — | | , ourerwise, 0. | | |



- [Description]
 (1) Logically left-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the lower 5 bits of general-purpose register reg1, by shifting "0" to LSB. The result is written to general-purpose register reg2. General-purpose register reg1 is not affected.
 (2) Logically left shifts the word data of general summers register reg2 by 'n' (0 to +21), the
 - (2) Logically left-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the 5-bit immediate data, zero-extended to word length, by shifting "0" to LSB. The result is written to general-purpose register reg2.
 - (3) Logically left-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the lower 5 bits of general-purpose register reg1, by shifting "0" to LSB. The result is written to general-purpose register reg3. General-purpose registers reg1 and reg2 are not affected.



<Data manipulation instruction>

| спр | | | Shift logical right by reg | ister/immediate (5-bit) |
|----------------------|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|-------------------------|
| ЭПК | | | | Logical right shift |
| [Instruction format] | (1) SHR | reg1, reg2 | | |
| | (2) SHR | imm5, reg2 | | |
| | (3) SHR | reg1, reg2, reg3 | | |
| [Operation] | (1) GR [1 | eg2] ← GR [reg2] logically shif | ft right by GR [reg1] | |
| | (2) GR [1 | $eg2] \leftarrow GR [reg2] \logically shifts a gradient of the set of the set$ | ft right by zero-extend (imm5) | |
| | (3) GR [1 | eg3] \leftarrow GR [reg2] logically shif | ft right by GR [reg1] | |
| [Format] | (1) Form | ıt IX | | |
| | (2) Form | t II | | |
| | (3) Form | ıt XI | | |
| [Opcode] | | | | |
| | (1) 15 | 031 | 16 01000000 | |
| | (2) 15 | 0 rr010100iiiii | | |
| | 15 (3) rr: | 031 | 16 010000010 | |
| [Flags] | | | | |
| - | CY | "1" if the last bit shifted out is "1 | 1"; otherwise, "0" including non-shift | |
| | OV S | 0 "1" if the operation result is nea | ative: otherwise "N" | |
| | z | "1" if the operation result is "0"; | otherwise, "0". | |
| | SAT | _ | | |



[Description] (1) Logically right-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the lower 5 bits of general-purpose register reg1, by shifting "0" to MSB. The result is written to general-purpose register reg2. General-purpose register reg1 is not affected.

- (2) Logically right-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the 5-bit immediate data, zero-extended to word length, by shifting "0" to MSB. The result is written to general-purpose register reg2.
- (3) Logically right-shifts the word data of general-purpose register reg2 by 'n' (0 to +31), the position specified by the lower 5 bits of general-purpose register reg1, by shifting "0" to MSB. The result is written to general-purpose register reg3. General-purpose registers reg1 and reg2 are not affected.



<Load instruction>

| SLD.B | Short format load I | byte |
|----------------------|-------------------------------------------------------------------------------------------|------|
| | Load of (signed) byte of | data |
| [Instruction format] | SLD.B disp7 [ep], reg2 | |
| [Operation] | adr \leftarrow ep + zero-extend (disp7) ^{*1} | |
| | GR [reg2] ← sign-extend (Load-memory (adr, Byte)) | |
| | Note 1. An MDP exception might occur depending on the result of address calculation. | |
| [Format] | Format IV | |
| [Opcode] | 15 0 rrrr0110dddddd | |
| [Flags] | CY – OV – S – Z – SAT – | |
| [Description] | Adds the 7-bit displacement data, zero-extended to word length, to the element pointer to | |

length, and stored in reg2.

generate a 32-bit address. Byte data is read from the generated address, sign-extended to word



| SLD.BU | Short format load byte unsigned |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Load of (unsigned) byte data |
| [Instruction format] | SLD.BU disp4 [ep], reg2 |
| [Operation] | adr ← ep + zero-extend (disp4) ^{*1} GR [reg2] ← zero-extend (Load-memory (adr, Byte)) |
| | Note 1. An MDP exception might occur depending on the result of address calculation. |
| [Format] | Format IV |
| [Opcode] | $\frac{15 \qquad 0}{rrrrr \neq 00000 \text{ (Do not specify r0 for reg2.)}}$ |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the 4-bit displacement data, zero-extended to word length, to the element pointer to generate a 32-bit address. Byte data is read from the generated address, zero-extended to word length, and stored in reg2. |
| | CAUTION |
| | Do not specify r0 for reg2. |



.

| SLD.H | Short format load halfword |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Load of (signed) halfword data |
| [Instruction format] | SLD.H disp8 [ep], reg2 |
| [Operation] | adr ← ep + zero-extend (disp8) ^{*1} GR [reg2] ← sign-extend (Load-memory (adr, Halfword)) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format IV |
| [Opcode] | 15 0 rrrrr1000dddddd |
| [Flags] | dddddd is the higher 7 bits of disp8. CY — OV — S — Z — SAT — |
| [Description] | Adds the element pointer to the 8-bit displacement data, zero-extended to word length, to generate a 32-bit address. Halfword data is read from this 32-bit address, sign-extended to word length, and stored in general-purpose register reg2. |



| SLD.HU | Short format load halfword unsigned Load of (unsigned) halfword data |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | SLD.HU disp5 [ep], reg2 |
| [Operation] | adr ← ep + zero-extend (disp5) ^{*1} GR [reg2] ← zero-extend (Load-memory (adr, Halfword)) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format IV |
| [Opcode] | $\frac{15 \qquad 0}{rrrrr 0000111dddd}$ rrrrr $\neq 00000$ (Do not specify r0 for reg2.) dddd is the higher 4 bits of disp5. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the element pointer to the 5-bit displacement data, zero-extended to word length, to generate a 32-bit address. Halfword data is read from this 32-bit address, zero-extended to word length, and stored in general-purpose register reg2. |
| | CAUTION |
| | Do not specify r0 for reg2. |



.

| SLD.W | Short format load word |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Load of word data |
| [Instruction format] | SLD.W disp8 [ep], reg2 |
| [Operation] | adr ← ep + zero-extend (disp8) ^{*1} GR [reg2] ← Load-memory (adr, Word) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format IV |
| [Opcode] | 15 0 rrrrr1010dddddd0 dddddd is the higher 6 bits of disp8. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the element pointer to the 8-bit displacement data, zero-extended to word length, to generate a 32-bit address. Word data is read from this 32-bit address, and stored in general-purpose register reg2. |



| SNOOZE | | Snooze |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------|
| | | Snooze |
| [Instruction format] | Snooze | |
| [Operation] | Snooze while hardware-defined period | |
| [Format] | Format X | |
| [Opcode] | 15 0 31 16 0000111111100000 000000100100000 000000000000000000000000000000000000 | |
| [Flags] | CY — OV — S — Z — SAT — | |
| [Description] | Temporarily halts operation of the CPU core for the period defined by the hardware specifications or when the CPU enters a specific state. When the specified period has elapsed or the CPU exits the specified state, CPU op automatically resumes and instruction execution begins from the next instruction. The SNOOZE state is released under the following conditions: | eration |
| | • The predefined period of time passes | |
| | • A terminating exception occurs | |
| | Even if the conditions for acknowledging the above exceptions are not satisfied (due or NP value), as long as a SNOOZE mode release request exists, the SNOOZE state released (for example, even if PSW.ID = 1, the SNOOZE state is released when INT Note, however, that the SNOOZE mode will not be released if terminating exception masked by the following mask settings, which are defined individually for each function. | to the ID is 0 occurs). ons are action: |
| | • Terminating exceptions are masked by an interrupt channel mask setting specifinterrupt controller* ¹ . | ied by the |
| | • Terminating exceptions are masked by a mask setting specified by using the fle point operation exception enable bit. | oating- |
| | • Terminating exceptions are masked by a mask setting defined by a hardware for other than the above. | inction |
| | Note 1. This does not include masking specified by the ISPR and PMR registers. | |

[Supplement]

This instruction is used to prevent the CPU performance from dropping in a multi-core system due to bus band occupancy during a spinlock.

CAUTION

The period of the pause triggered by the SNOOZE instruction is defined according to the hardware specifications of the CPU core. For details, see the hardware manual of the product used.



| SST.B | | Short format store byte |
|----------------------|--------------------------------------------------------------------------------------|-------------------------------------------------|
| | | Storage of byte data |
| [Instruction format] | SST.B reg2, disp7 [ep] | |
| [Operation] | adr ← ep + zero-extend (disp7) ^{*1} Store-memory (adr, GR [reg2] , Byte) | |
| | Note 1. An MDP exception might occur | depending on the result of address calculation. |
| [Format] | Format IV | |
| [Opcode] | 15 0 rrrrr0111ddddddd | |
| [Flags] | CY — OV — S — Z — SAT — | |
| [Description] | Adds the element pointer to the 7-bit disp | lacement data, zero-extended to word length, to |

generate a 32-bit address and stores the data of the lowest byte of reg2 to the generated

address.



.

| SST.H | Short format store halfword |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Storage of halfword data |
| [Instruction format] | SST.H reg2, disp8 [ep] |
| [Operation] | adr ← ep + zero-extend (disp8) ^{*1} Store-memory (adr, GR [reg2], Halfword) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format IV |
| [Opcode] | 15 0 rrrr1001dddddd dddddd is the higher 7 bits of disp8. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the element pointer to the 8-bit displacement data, zero-extended to word length, to generate a 32-bit address, and stores the lower halfword data of reg2 to the generated 32-bit address. |



| SST.W | Short format store word |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Storage of word data |
| [Instruction format] | SST.W reg2, disp8 [ep] |
| [Operation] | adr ← ep + zero-extend (disp8) ^{*1} Store-memory (adr, GR [reg2], Word) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format IV |
| [Opcode] | 15 0 rrrrr1010ddddd1 ddddd is the higher 6 bits of disp8. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the element pointer to the 8-bit displacement data, zero-extended to word length, to generate a 32-bit address and stores the word data of reg2 to the generated 32-bit address. |



| ST.B | | | | | | | | Store byte |
|----------------------|----------------------|-------------------------------------------------|--------------------------------------------------|-----------------------------------------------------|-----------------------------------------|----------------------------------------------|---------------------------------|-------------------------|
| | | | | | | | Storage | e of byte data |
| [Instruction format] | (1) (2) | ST.B reg2, di ST.B reg3, di | sp16 [reg1] sp23 [reg1] | | | | | |
| [Operation] | (1) | adr ← GR [re Store-memor | g1] + sign-e: y (adr, GR [r | xtend (disp16) eg2], Byte) |)*1 | | | |
| | (2) | adr ← GR [re Store-memor | g1] + sign-e y (adr, GR [r | xtend (disp23) eg3], Byte) |)*1 | | | |
| | | Note 1. An MI | DP exception r | night occur dep | ending on t | he result of add | ress calculatior | ı. |
| [Format] | (1) (2) | Format VII Format XIV | | | | | | |
| [Opcode] | | 15 | 0 | 31 | 1 | 6 | | |
| | (1) | rrrrr111 | 010RRRRR | dddddddd | dddddd | A | | |
| | | 15 | 0 | 31 | 1 | 6 47 | 32 | |
| | (2) | 00000111 | 100RRRRR | wwwwwdddd | ddd1101 | | DDDDDDDD | |
| | When dddo DDDI | re RRRRR = r dddd is the lo DDDDDDDDDD | eg1, wwwww ower 7 bits of DDDD is the l | = reg3. f disp23. nigher 16 bits | of disp23 | | | |
| [Flags] | СҮ | _ | | | | | | |
| | OV | _ | | | | | | |
| | S | — | | | | | | |
| | SAT | _ | | | | | | |
| [Description] | (1) | Adds the data extended to w general-purpo | of general-p ord length, t ose register re | ourpose registe o generate a 3 eg2 to the gen | er reg1 to 32-bit addr erated add | the 16-bit disp ress and stores lress. | placement data the lowest by | a, sign- yte data of |
| | (2) | Adds the data extended to w general-purpo | of general-p ord length, t ose register re | ourpose registe o generate a 3 eg3 to the gen | er reg1 to 32-bit addr erated add | the 23-bit disp ress and stores Iress. | blacement data | a, sign- yte data of |

.

| ST.DW | Store Double Word |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | ST DW reg3_disp23[reg1] |
| | |
| [Operation] | adr \leftarrow GR [reg1] + sign-extend (disp23) ^{*1} data \leftarrow GR[reg3+1] GR[reg3] Store-memory (adr, data, Double-word) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format XIV |
| [Opcode] | 15 031 1647 32 |
| | Where RRRRR = reg1, wwwww = reg3. dddddd is the lower side bits 6 to 1 of disp23. DDDDDDDDDDDDDDDD is the higher 16 bits of disp23. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Adds the data of general-purpose register reg1 to a 23-bit displacement value sign-extended to word length to generate a 32-bit address. Doubleword data consisting of the lower 32 bits of the word data of general-purpose register reg3 and the higher 32 bits of the word data of reg3 + 1 is then stored at this address. |
| [Supplement] | reg3 must be an even-numbered register. |



| ST.H | Store halfwore |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| | Storage of halfword data |
| [Instruction format] | (1) ST.H reg2, disp16 [reg1] (2) ST.H reg3, disp23 [reg1] |
| [Operation] | (1) adr \leftarrow GR [reg1] + sign-extend (disp16) ^{*1} Store memory (adr. GP [reg2] Helfword) |
| | (2) adr ← GR [reg1] + sign-extend (disp23)*1 Store-memory (adr, GR [reg3], Halfword) |
| | Note 1. An MAE or MDP exception might occur depending on the result of address calculation. |
| [Format] | Format VII Format XIV |
| [Opcode] | 15 031 16 |
| | (1) rrrr111011RRRRR dddddddddddddd |
| | Where dddddddddddd is the higher 15 bits of disp16. |
| | 15 031 1647 32 (2) 00000111101RRRRR wwwwwdddddd01101 DDDDDDDDDDDDDDDDDDDD |
| | Where RRRRR = reg1, wwwww = reg3. dddddd is the lower side bits 6 to 1 of disp23. DDDDDDDDDDDDDDDDD is the higher 16 bits of disp23. |
| [Flags] | CY — |
| | ov — |
| | S — |
| | Z — SAT — |
| | |



[Description]

- (1) Adds the data of general-purpose register reg1 to the 16-bit displacement data, signextended to word length, to generate a 32-bit address and stores the lower halfword data of general-purpose register reg2 to the generated address.
- (2) Adds the data of general-purpose register reg1 to the 23-bit displacement data, signextended to word length, to generate a 32-bit address and stores the lower halfword data of general-purpose register reg3 to the generated address.



| | | | | | | | Store word |
|----------------------|------------------------------------------|------------------------------------------|-------------------------------------------|-------------------------------------------------|-----------------|-----------------------------|----------------|
| 51.00 | | | | | | Storag | e of word data |
| [Instruction format] | (1) S (2) S | T.W reg2, di T.W reg3, di | sp16 [reg1] sp23 [reg1] | | | | |
| [Operation] | (1) a S | dr ← GR [reg tore-memory | g1] + sign-e: (adr, GR [r | xtend (disp16) [*] eg2], Word) | 1 | | |
| | (2) a S | dr ← GR [reg tore-memory | g1] + sign-e: (adr, GR [re | xtend (disp23) [*] eg3], Word) | 1 | | |
| | N | iote 1. An MA | E or MDP exc | ception might occ | ur depending | on the result of address of | calculation. |
| [Format] | (1) F (2) F | ormat VII ormat XIV | | | | | |
| [Opcode] | | 15 | 0 | 31 | 16 | | |
| | (1) | rrrr1110 | 11RRRRR | ddddddddd | ddddd1 | | |
| | Where | e dddddddd | dddddd i | s the higher 15 | bits of disp | 16. | |
| | | 15 | 0 | 31 | 16 47 | 7 32 | |
| | (2) | 000001111 | 00RRRRR | wwwwwddddd | d01111 D | | |
| | Where ddddo DDDD | RRRRR = re dd is the low DDDDDDDDD | g1, wwwww er side bits DDD is the l | = reg3. 6 to 1 of disp23 nigher 16 bits o | s. f disp23. | | |
| [Flags] | СҮ | _ | | | | | |
| | OV | _ | | | | | |
| | S - | — | | | | | |
| | Z | — | | | | | |

SAT

_



[Description]

- (1) Adds the data of general-purpose register reg1 to the 16-bit displacement data, signextended to word length, to generate a 32-bit address and stores the word data of generalpurpose register reg2 to the generated 32-bit address.
- (2) Adds the data of general-purpose register reg1 to the 23-bit displacement data, signextended to word length, to generate a 32-bit address and stores the word data of generalpurpose register reg3 to the generated 32-bit address.



| | Store Conditional | | | | | |
|----------------------|--------------------------------------------------------------------------------------------------|--|--|--|--|--|
| STC.W | | | | | | |
| | Conditional storage when atomic word data manipulation is complete | | | | | |
| [Instruction format] | STC.W reg3, [reg1] | | | | | |
| [Operation] | adr ← GR[reg1] ^{*1} | | | | | |
| | data \leftarrow GR[reg3] | | | | | |
| | token ← LLbit ^{*2} | | | | | |
| | if $(token == 1)$ | | | | | |
| | then Store-memory (adr, data, Word) | | | | | |
| | $GR[reg3] \leftarrow 1$ | | | | | |
| | else GR[reg3] $\leftarrow 0$ | | | | | |
| | endif | | | | | |
| | LLbit $\leftarrow 0^{*2}$ | | | | | |
| | | | | | | |
| | Note 1. An MAE, MDP exception might occur depending on the result of address calculation. | | | | | |
| | Using the LDL.W and STC.W Instructions. | | | | | |
| [Format] | Format VII | | | | | |
| [Oncoda] | | | | | | |
| [Opcode] | 15 031 16 | | | | | |
| | 00000111111RRRRR wwwww01101111010 | | | | | |
| | | | | | | |
| [Flags] | | | | | | |
| | | | | | | |
| | S — | | | | | |
| | Z — | | | | | |
| | SAT — | | | | | |
| | | | | | | |



| [Description] | This instruction can only be executed successfully if a link exists that corresponds to the | | | | | |
|---------------|-------------------------------------------------------------------------------------------------|--|--|--|--|--|
| | specified address. If a corresponding link exists, the word data of general-purpose register | | | | | |
| | reg3 is stored in the memory and an atomic read-modify-write is executed. | | | | | |
| | If the corresponding link has been lost, the data is not stored in the memory and execution of | | | | | |
| | this instruction fails. | | | | | |
| | Whether execution of the STC.W instruction has succeeded or not can be ascertained by | | | | | |
| | checking the contents of general-purpose register reg3 after the instruction has been executed. | | | | | |
| | If execution of the STC.W instruction was successful, general-purpose register reg3 will be | | | | | |
| | set (1). If execution failed, reg3 will be cleared (0). | | | | | |
| | This instruction can be used together with the LDL.W instruction to ensure accurate updating | | | | | |
| | of the memory in a multi-core system. | | | | | |
| | | | | | | |
| [Supplement] | Use the LDL.W and STC.W instructions instead of the CAXI instruction if an atomic | | | | | |
| | guarantee is required when updating the memory in a multi-core system. | | | | | |



| STSR | Store contents of system register |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Storage of contents of system register |
| [Instruction format] | STSR regID, reg2, selID STSR regID, reg2 |
| [Operation] | $GR [reg2] \leftarrow SR [regID, selID]^{*1}$ |
| | Note 1. An exception might occur depending on the access permission. For details, see Section 2.5.3 , Register Updating . |
| [Format] | Format IX |
| [Opcode] | 15 031 16 rrrrr111111RRRR sssss00001000000 rrrrr: reg2, sssss: selID, RRRR: regID |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Stores the system register contents specified by the system register number and group number (regID, selID) in general-purpose register reg2. The system register is not affected. If selID is omitted, it is assumed that selID is 0. |
| [Supplement] | A PIE or UCPOP exception might occur as a result of executing this instruction, depending on the combination of CPU operating mode and system register to be accessed. For details, see Section 2.5.3, Register Updating . |
| | CAUTION |
| | The system register number or group number is a unique number used to identify each system register. How to access undefined registers is described in Section 2.5.4 , Accessing Undefined Registers , but accessing undefined registers is not recommended. |



<Arithmetic instruction>

| SUB | | Subtract |
|----------------------|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Subtraction |
| [Instruction format] | SUB reg | 1, reg2 |
| [Operation] | GR [reg2 | $] \leftarrow GR [reg2] - GR [reg1]$ |
| [Format] | Format I | |
| [Opcode] | 15 rrrr0 | 0 01101RRRRR |
| [Flags] | CY OV S Z SAT | "1" if a borrow occurs from MSB; otherwise, "0". "1" if overflow occurs; otherwise, "0". "1" if the operation result is negative; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". |
| [Description] | Subtracts purpose r register re | the word data of general-purpose register reg1 from the word data of general- egister reg2 and stores the result in general-purpose register reg2. General-purpose eg1 is not affected. |



<Arithmetic instruction>

| SUBR | | Subtract reverse |
|----------------------|---------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Reverse subtraction |
| [Instruction format] | SUBR re | .g1, reg2 |
| [Operation] | GR [reg2 | $] \leftarrow GR [reg1] - GR [reg2]$ |
| [Format] | Format I | |
| [Opcode] | 15 rrrr0 | 0 01100RRRR |
| [Flags] | CY OV S Z SAT | "1" if a borrow occurs from MSB; otherwise, "0". "1" if overflow occurs; otherwise, "0". "1" if the operation result is negative; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". |
| [Description] | Subtracts purpose r register re | the word data of general-purpose register reg2 from the word data of general- egister reg1 and stores the result in general-purpose register reg2. General-purpose eg1 is not affected. |



٦

| SWITCH | Jump with table look up | | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| [Instruction format] | SWITCH reg1 | | |
| [Operation] | adr \leftarrow (PC + 2) + (GR [reg1] logically shift left by 1) ^{*1} PC \leftarrow (PC + 2) + (sign-extend (Load-memory (adr, Halfword))) logically shift left by 1 | | |
| | Note 1. An MDP exception might occur depending on the result of address calculation. | | |
| [Format] | Format I | | |
| [Opcode] | 15 0 000000010RRRR | | |
| | RRRRR \neq 00000 (Do not specify r0 for reg1.) | | |
| [Flags] | CY – OV – S – Z – SAT – | | |
| [Description] | The following steps are taken. (1) Adds the start address (the one subsequent to the SWITCH instruction) to general-purpose register reg1, logically left-shifted by 1, to generate a 32-bit table entry address. (2) Loads the halfword entry data indicated by the address generated in step (1). (3) Adds the table start address after sign-extending the loaded halfword data and logically left-shifting it by 1 (the one subsequent to the SWITCH instruction) to generate a 32-bit target address. (4) Jumps to the target address generated in step (3). | | |
| | CAUTIONS | | |
| | Do not specify r0 for reg1. In the SWITCH instruction memory read operation executed in order to read the table, memory protection is performed. | | |



<Data manipulation instruction>

| SXB | | Sign extend b | yte |
|----------------------|---------------------------|-----------------------------------------------------------------------|-----|
| | | Sign-extension of byte d | ata |
| [Instruction format] | SXB reg | 1 | |
| [Operation] | GR [reg1] |] \leftarrow sign-extend (GR [reg1] (7:0)) | |
| [Format] | Format I | | |
| [Opcode] | 15 0000000 | 0 00101RRRRR | |
| [Flags] | CY OV S Z SAT | | |
| [Description] | Sign-exter | ends the lowest byte of general-purpose register reg1 to word length. | |



<Data manipulation instruction>

| SXH | | Sign extend halfword |
|----------------------|-------------------------------------|--------------------------------------------------------------|
| | | Sign-extension of halfword data |
| [Instruction format] | SXH reg1 | |
| [Operation] | GR [reg1] ← sign-ex | tend (GR [reg1] (15:0)) |
| [Format] | Format I | |
| [Opcode] | 15 00000000111RRR | 0 RR |
| [Flags] | CY – OV – S – Z – SAT – | |
| [Description] | Sign-extends the low | er halfword of general-purpose register reg1 to word length. |



| SYNCE | | | Synchronize exceptions | |
|----------------------|---------------------------|-----------------------------------|---------------------------------------|--|
| | | | Exception synchronization instruction | |
| [Instruction format] | SYNCE | | | |
| [Operation] | No operat | tion is performed. | | |
| [Format] | Format I | | | |
| [Opcode] | 15 0000000 | 0 0000011101 | | |
| [Flags] | CY OV S Z SAT | | | |
| [Description] | In this CP | PU, the SYNCE instruction is hand | lled as the NOP instruction. | |



| | Synchronize instruction pipeline | | |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| SYNCI | | | |
| | Instruction pipeline synchronization instruction | | |
| [Instruction format] | SYNCI | | |
| [Operation] | Resolves instruction hazards. | | |
| [Format] | Format I | | |
| [Opcode] | 15 0 0000000011100 | | |
| [Flags] | CY – OV – S – Z – SAT – | | |
| [Description] | Makes subsequent instructions wait until all the instructions ahead of this instruction have finished executing. The instructions executed after the SYNCI instruction are guaranteed to adapt to the effects produced by the execution of the instructions preceding SYNCI. This instruction can be used to realize "self-programming code" to overwrite instructions in the memory. | | |
| [Supplement] | The SYNCI instruction clears the CPU instruction fetch pipeline so that subsequently executed instructions are re-fetched. If the CPU includes an instruction cache, the instruction cache must be disabled to enable the realization of this self-programming code. | | |



| | Synchronize memory | |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| STNCM | Memory synchronize instruction | |
| [Instruction format] | SYNCM | |
| [Operation] | Waits for the synchronization of the memory device. | |
| [Format] | Format I | |
| [Opcode] | 15 0 0000000011110 | |
| [Flags] | CY – OV – S – Z – SAT – | |
| [Description] | Synchronizes the CPU execution pipeline and memory accesses. Waits for the start of execution until all preceding memory access has been synchronized. "Synchronization" refers to the status where the result of preceding memory accesses can be referenced by any master device within the system. In cases such as when buffering is used to delay memory accesses and synchronization of all memory accesses has not occurred, the SYNCM instruction does not complete and waits for the synchronization. The subsequent instructions will not be executed until the SYNCM instruction execution is complete. This instruction can be used to realize the "synchronization primitive" in a multi-processing environment, if the above function is provided as the system. | |



| SYNCP | | | Synchronize pipeline |
|----------------------|---------------------------|-------------------------------------------------|----------------------------------|
| | | | Pipeline synchronize instruction |
| [Instruction format] | SYNCP | | |
| [Operation] | Waits for | the synchronization of pipeline. | |
| [Format] | Format I | | |
| [Opcode] | 15 0000000 | 0 0000011111 | |
| [Flags] | CY OV S Z SAT | | |
| [Description] | Waits unti | il execution of all previous instructions is co | ompleted before being executed. |


<Special instruction>

| evec ALL | System call |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SIJUALL | System call exception |
| | |
| [Instruction format] | SYSCALL vector8 |
| [Operation] | $EIPC \leftarrow PC + 4$ (return PC) |
| | $EIPSW \leftarrow PSW$ |
| | EIIC \leftarrow exception cause code ^{*1} |
| | $\text{PSW.UM} \leftarrow 0$ |
| | $PSW.EP \leftarrow 1$ |
| | $PSW.ID \leftarrow 1$ |
| | if (vector8 <= SCCFG.SIZE) is satisfied |
| | then $adr \leftarrow SCBP + zero-extend (vector8 logically shift left by 2)^2$ else $adr \leftarrow SCBP^{*2}$ |
| | $PC \leftarrow SCBP + Load-memory (adr, Word)$ |
| | Note 1. See Table 4.1, Exception Cause List.Note 2. An MDP exception might occur depending on the result of address calculation. |
| [Format] | Format X |
| [Opcode] | 15 0.21 16 |
| | 11010111111vvvvv 00vvv00101100000 |
| | Where VVV is the higher 3 bits of vector8 and vvvvv is the lower 5 bits of vector8. |
| [Flags] | |
| | CY — |
| | ov — |
| | s — |
| | Z — |
| | SAT — |



[Description]

- (1) Saves the contents of the return PC (address of the instruction next to the SYSCALL instruction) and PSW to EIPC and EIPSW.
- (2) Stores the exception cause code corresponding to vector8 in the EIIC register. The exception cause code is the value of vector8 plus 8000_H.
- (3) Updates the PSW according to the exception causes listed in **Table 4.1**.
- (4) Generates a 32-bit table entry address by adding the value of the SCBP register and vector8 that is logically shifted 2 bits to the left and zero-extended to a word length. If vector8 is greater than the value specified by the SIZE bit of system register SCCFG; however, vector8 that is used for the generation of a 32-bit table entry address is handled as 0.
- (5) Loads the word of the address generated in (4).
- (6) Generates a 32-bit target address by adding the value of the SCBP register to the data in (5).
- (7) Branches to the target address generated in (6).

CAUTION

In the SYSCALL instruction memory read operation executed in order to read the table, memory protection is performed with the supervisor privilege.



<Special instruction>

| ТРАР | | Тгар |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| | | Software exception |
| [Instruction format] | TRAP vector5 | |
| [Operation] | EIPC \leftarrow PC + 4 (return PC) EIPSW \leftarrow PSW EIIC \leftarrow exception cause code ^{*1} PSW.UM \leftarrow 0 PSW.EP \leftarrow 1 PSW.ID \leftarrow 1 PC \leftarrow exception handler address ^{*2} | |
| | Note 1. See Table 4.1, Exception Cause List. Note 2. See Section 4.4, Exception Handler Address. | |
| [Format] | Format X | |
| [Opcode] | 15 031 16 00000111111vvvvv 0000000000000 vvvvv = vector5 | |
| [Flags] | CY — OV — S — Z — SAT — | |



[Description]

Saves the contents of the return PC (address of the instruction next to the TRAP instruction) and the current contents of the PSW to EIPC and EIPSW, respectively, stores the exception cause code in the EIIC register, and updates the PSW according to the exception causes listed in **Table 4.1**. Execution then branches to the exception handler address and exception handling is started.

The following table shows the correspondence between vector5 and exception cause codes and exception handler address offset. Exception handler addresses are calculated based on the offset addresses listed in the following table. For details, see **Section 4.4, Exception Handler Address**.

| vector5 | Exception Cause Code | Offset Address |
|-----------------|------------------------|-----------------|
| 00 _H | 0000 0040 _H | 40 _H |
| 01 _H | 0000 0041 _H | |
| | | |
| 0F _H | 0000 004F _H | |
| 10 _H | 0000 0050 _H | 50 _H |
| 11 _H | 0000 0051 _H | |
| | | |
| 1F _H | 0000 005F _H | |



<Logical instruction>

| тѕт | | | Test Test |
|----------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| [Instruction format] | TST reg | l, reg2 | |
| [Operation] | result ← | GR [reg2] AND GR [reg1] | |
| [Format] | Format I | | |
| [Opcode] | 15 rrrrr0(| 0 01011RRRRR | |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, 0. — | |
| [Description] | ANDs the register re registers 1 | e word data of general-purpose register reg2 with the word data of general-purp eg1. The result is not stored with only the flags being changed. General-purpose reg1 and reg2 are not affected. | ose |



<Bit manipulation instruction>

| TST1 | | Test bit |
|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|
| | | Bit test |
| [Instruction format] | (1) TST1 bit#3, disp16 [reg1] | |
| | (2) TST1 reg2, [reg1] | |
| [Operation] | adr ← GR [reg1] + sign-extend (disp16)^{*1} token ← Load-memory (adr, Byte) Z flag ← Not (extract-bit (token, bit#3)) | |
| | (2) $adr \leftarrow GR [reg1]^{*1}$ token \leftarrow Load-memory (adr, Byte) Z flag \leftarrow Not (extract-bit (token, reg2)) | |
| | Note 1. An MDP exception might occur depending on the result of address calculation. | |
| [Format] | (1) Format VIII | |
| | (2) Format IX | |
| [Opcode] | | |
| | | |
| | (1) 11bbb111110RRRRR ddddddddddddddddd | |
| | <u>15 0 31 16</u> | |
| | (2) rrrr111111RRRRR 000000011100110 | |
| | | |
| [Flags] | Сү — | |
| | OV — | |
| | S — Z "1" if bit specified by operand = "0", "0" if bit specified by operand = "1" | |
| | SAT — | |



- [Description]
 (1) Adds the word data of general-purpose register reg1 to the16-bit displacement data, sign-extended to word length, to generate a 32-bit address; checks the bit specified by the 3-bit bit number at the byte data location referenced by the generated address. If the specified bit is "0", "1" is set to the Z flag of PSW and if the bit is "1", the Z flag is cleared to "0". The byte data, including the specified bit, is not affected.
 - (2) Reads the word data of general-purpose register reg1 to generate a 32-bit address; checks the bit specified by the lower 3 bits of reg2 at the byte data location referenced by the generated address. If the specified bit is "0", "1" is set to the Z flag of PSW and if the bit is "1", the Z flag is cleared to "0". The byte data, including the specified bit, is not affected.



<Logical instruction>

| XOR | | | Exclusive OR |
|----------------------|-------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|
| | | | Exclusive OR |
| [Instruction format] | XOR reg | 1, reg2 | |
| [Operation] | GR [reg2 |] \leftarrow GR [reg2] XOR GR [reg1] | |
| [Format] | Format I | | |
| [Opcode] | 15 rrrr0(| 0 01001RRRRR | |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". — | |
| [Description] | Exclusive general-p purpose r | ely ORs the word data of general-purpose register reg2 with the wor urpose register reg1 and stores the result in general-purpose register egister reg1 is not affected. | rd data of r reg2. General- |



<Logical instruction>

| XORI | | Exclusive OR immediate (16-bit) |
|----------------------|--------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | | Exclusive OR immediate |
| [Instruction format] | XORI im | nm16, reg1, reg2 |
| [Operation] | GR [reg2 |] \leftarrow GR [reg1] XOR zero-extend (imm16) |
| [Format] | Format V | 1 |
| [Opcode] | 15 rrrrl: | 031 16 10101RRRRR iiiiiiiiiiiiii |
| [Flags] | CY OV S Z SAT | — 0 "1" if operation result word data MSB is "1"; otherwise, "0". "1" if the operation result is "0"; otherwise, "0". — |
| [Description] | Exclusive data, zero General-r | ely ORs the word data of general-purpose register reg1 with the 16-bit immediate p-extended to word length, and stores the result in general-purpose register reg2. |



<Data manipulation instruction>

| ZXB | | | Zero extend byte |
|----------------------|---------------------------|------------------------------------------------|-----------------------------|
| | | | Zero-extension of byte data |
| [Instruction format] | ZXB reg | 1 | |
| [Operation] | GR [reg1] |] \leftarrow zero-extend (GR [reg1] (7:0)) | |
| [Format] | Format I | | |
| [Opcode] | 15 0000000 | 0 D0100RRRRR | |
| [Flags] | CY OV S Z SAT | | |
| [Description] | Zero-exte | nds the lowest byte of general-purpose registe | er reg1 to word length. |



<Data manipulation instruction>

| zхн | | | Zero extend halfword |
|----------------------|---------------------------|-----------------------------------------------|---------------------------------|
| | | | Zero-extension of halfword data |
| [Instruction format] | ZXH reg1 | | |
| [Operation] | GR [reg1] | \leftarrow zero-extend (GR [reg1] (15:0)) | |
| [Format] | Format I | | |
| [Opcode] | 15 0000000 | 0 0110RRRRR | |
| [Flags] | CY OV S Z SAT | | |
| [Description] | Zero-exten | ids the lower halfword of general-purpose reg | ister reg1 to word length. |



7.3 Cache Instructions

7.3.1 Overview of Cache Instructions

This CPU provides the cache instructions to enable efficient manipulation of the cache by the CPU.

The following cache instructions (mnemonics) are available.

- CACHE: Cache
- PREF: Prefetch

7.3.2 Cache Instruction Set

This section details each instruction, dividing each mnemonic (in alphabetical order) into the following items.

- Instruction format: Indicates how the instruction is written and its operand(s).
- Operation: Indicates the function of the instruction.
- Format: Indicates the instruction format.
- Opcode: Indicates the bit field of the instruction opcode.
- Description: Describes the operation of the instruction.
- Supplement: Provides supplementary information on the instruction.



<Cache instruction>

| CACHE | Cache Cache operation |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | CACHE cacheop, [reg1] |
| [Operation] | Manipulates the cache specified by cacheop. |
| [Format] | Format X |
| [Opcode] | 1503116111pp111111RRRRRPPPPP00101100000ppPPPPP indicates cacheop. |
| [Flags] | CY – OV – S – Z – SAT – |
| [Description] | Sets the word data of general-purpose register reg1 as a 32-bit address or the cache index and manipulates the cache specified by cacheop. For details about the cache index specification method, see Section 5.2.5, Cache Index Specification Method . |
| [Supplement] | Each cache operation has its own instruction execution privilege. For details about the correspondence between cache operations and instruction execution privileges, see Section 5.2.6, Execution Privilege of the CACHE/PREF Instruction . |
| | When manipulating the cache by specifying the address, it might become the target of memory protection by the MPU. For details about the relationship between cache manipulation and memory protection, see Section 5.2.7, Memory Protection for CACHE and PREF Instructions. |



| cacheop | Target | Processing | Cache Specification | Operation |
|---------|-------------|------------|---------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0000000 | Instruction | СНВІІ | Address | (Cache Hit Block Invalidate, Instruction cache) If the specified address hits an address in the instruction cache, the corresponding cache line is disabled. If the specified address does not hit an address in the instruction cache, no processing is performed. |
| 0100000 | Instruction | CIBII | Index | (Cache Indexed Block Invalidate, Instruction cache) Disables the instruction cache line of the specified index. This instruction can be used in cases such as when the entire memory cache is initialized by software. |
| 1000000 | Instruction | CFALI | Address | (Cache Fetch And Lock, Instruction cache) Loads the data from the specified address and stores it in the instruction cache. At this time, the corresponding cache line is locked. If the data at the specified address is already stored in the instruction cache, this instruction only locks the cache line. If the data at the specified address is already stored in the instruction cache and the corresponding cache line is locked, no processing is performed. |
| 1100000 | Instruction | CISTI | Index | (Cache Indexed Store, Instruction cache) Writes (stores) data from a system register to the instruction cache line of the specified index. The specifications of the data to be written and the system register depend on the specifications of the CPU core. For details, see the hardware manual of the product used. |
| 1100001 | Instruction | CILDI | Index | (Cache Indexed Load, Instruction cache) Reads (loads) data from the instruction cache line of the specified index to a system register. The specifications of the data to be read and the system register depend on the specifications of the CPU core. For details, see the hardware manual of the product used. |
| 1111110 | _ | CLL | _ | (Clear Load Link-bit) Functions as the CLL instruction. |

| Table 7 7 | Cache | Operation |
|-----------|-------|-----------|
| | Gaune | operation |



<Cache instruction>

| PREF | P | refetch |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------|
| | Р | refetch |
| [Instruction format] | PREF prefop, [reg1] | |
| [Operation] | Executes the prefetch operation specified by prefop. | |
| [Format] | Format X | |
| [Opcode] | 15 031 16 1101111111RRRRR PPPP00101100000 | |
| | PPPPP indicates prefop. | |
| [Flags] | CY — OV — S — Z — SAT — | |
| [Description] | Executes the prefetch operation specified by prefop on the word data of general-purpos register reg1 used as a 32-bit address. | e |
| [Supplement] | The prefetch instruction does not generate a privilege instruction exception in any CPU in If the CPU being used does not have a cache, this instruction will not generate a reserve instruction exception and no processing is performed, in the same way as a NOP instruc- | mode. ed ction. |
| | Memory protection by the MMU does not generate memory protection exceptions (priv or access permission violation exceptions) during prefetch operations. In such a case, the prefetch operation is implicitly ignored and no processing is performed, in the same way NOP instruction. | vilege ne ny as a |
| | CAUTION | |
| | Be aware that even after the prefetch instruction has finished executing, a prefetch oper- might not necessarily have been performed. | ation |



| | Table 7.8 Prefetch Operat p Target Processing Cache Specificat 0 Instruction PREFI Address | | Prefetch Operation | |
|--------|------------------------------------------------------------------------------------------------------------------------------------|------------|---------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| prefop | Target | Processing | Cache Specification | Operation |
| 00000 | Instruction | PREFI | Address | (Prefetch Instruction cache) Stores the data at the specified address in the instruction cache. If the data at the specified address is already stored in the instruction cache, no processing is performed. |

CAUTION

The size of the data prefetched in one prefetch operation depends on the specifications of the CPU core.



7.4 Floating-Point Instructions

7.4.1 Instruction formats

All floating-point instructions are in 32-bit format.

When an instruction is actually saved to memory, it is placed as shown below.

- Lower part of instruction format (including bit $0) \rightarrow$ Lower address side
- Higher part of instruction format (including bit 15 or bit 31) \rightarrow Upper address side

(1) Format F:I

The 32-bit long floating-point instruction format includes a 6-bit opcode field, 4-bit subopcode field, three fields that specify general-purpose registers, a 3-bit category field, and a 2bit type field.

| 15 11 | 10 5 | 4 0 | 31 27 | 26 | 16 |
|-------|--------|------|-------|------------|----|
| | | | | | |
| reg2 | opcode | reg1 | reg3 | sub-opcode | |
| | | | | | |



7.4.2 Overview of Floating-Point Instructions

Floating-point instructions are divided into single-precision instructions (single) and double-precision instructions (double), and include the following instructions (mnemonics).

(1) Basic operation instructions

- ABSF.D: Floating-point Absolute Value (Double)
- ABSF.S: Floating-point Absolute Value (Single)
- ADDF.D: Floating-point Add (Double)
- ADDF.S: Floating-point Add (Single)
- DIVF.D: Floating-point Divide (Double)
- DIVF.S: Floating-point Divide (Single)
- MAXF.D: Floating-point Maximum (Double)
- MAXF.S: Floating-point Maximum (Single)
- MINF.D: Floating-point Minimum (Double)
- MULF.S: Floating-point Multiply (Single)
- NEGF.D: Floating-point Negate (Double)
- NEGF.S: Floating-point Negate (Single)
- RECIPF.D: Reciprocal of a floating-point value (Double)
- RECIPF.S: Reciprocal of a floating-point value (Single)
- RSQRTF.D: Reciprocal of the square root of a floating-point value (Double)
- RSQRTF.S: Reciprocal of the square root of a floating-point value (Single)
- SQRTF.D: Floating-point Square Root (Double)
- SQRTF.S: Floating-point Square Root (Single)
- SUBF.D: Floating-point Subtract (Double)
- SUBF.S: Floating-point Subtract (Single)

(2) Extended basic operation instructions

- FMAF.S: Floating-point fused-multiply-add (Single)
- FMSF.S: Floating-point fused-multiply-subtract (Single)
- FNMAF.S: Floating-point fused-negate-multiply-add (Single)
- FNMSF.S: Floating-point fused-negate-multiply-subtract (Single)

(3) Conversion instructions

- CEILF.SUW: Floating-point convert single to unsigned-word, round toward positive (Single)
- CEILF.DW: Floating-point Convert Double to Word, round toward positive (Double)
- CEILF.DUL: Floating-point Convert Double to Unsigned-Long, round toward positive (Double)
- CEILF.DUW: Floating-point Convert Double to Unsigned-Word, round toward positive (Double)
- CEILF.SL: Floating-point Convert Single to Long, round toward positive (Single)
- CEILF.SW: Floating-point Convert Single to Word, round toward positive (Single)
- CEILF.SUL: Floating-point Convert Single to Unsigned-Long, round toward positive (Single)
- CEILF.SUW: Floating-point Convert Single to Unsigned-Word, round toward positive (Single)
- CVTF.DL: Floating-point convert double to long (Double)
- CVTF.DS: Floating-point convert double to single (Double)
- CVTF.DUL: Floating-point convert double to unsigned-long (Double)
- CVTF.DUW: Floating-point convert double to unsigned-word (Double)
- CVTF.DW: Floating-point convert double to long (Double)
- CVTF.LD: Floating-point convert long to double (Double)
- CVTF.LS: Floating-point convert long to single (Single)
- CVTF.SD: Floating-point convert single to double (Double)
- CVTF.SL: Floating-point convert single to long (Single)
- CVTF.SUL: Floating-point convert single to unsigned-long (Single)
- CVTF.SUW: Floating-point convert single to unsigned-word (Single)
- CVTF.SW: Floating-point convert single to long (Single)
- CVTF.ULD: Floating-point convert unsigned-long to double (Double)
- CVTF.ULS: Floating-point convert unsigned-long to single (Single)
- CVTF.UWD: Floating-point convert unsigned-word to double (Double)
- CVTF.UWS: Floating-point convert unsigned-word to single (Single)
- CVTF.WD: Floating-point convert word to double (Double)
- CVTF.WS: Floating-point convert word to single (Single)
- FLOORF.DL: Floating-point convert double to long, round toward negative (Double)
- FLOORF.DW: Floating-point convert double to long, round toward negative (Double)
- FLOORF.DUL: Floating-point convert double to unsigned-long, round toward negative (Double)
- FLOORF.DUW: Floating-point convert double to unsigned-word, round toward

negative (Double)

- FLOORF.SL: Floating-point convert single to long, round toward negative (Single)
 - FLOORF.SW: Floating-point convert single to long, round toward negative (Single)
- FLOORF.SUL: Floating-point convert single to unsigned-long, round toward negative (Single)
- FLOORF.SUW: Floating-point convert single to unsigned-word, round toward negative (Single)
- TRNCF.DL: Floating-point convert double to long, round toward zero (Double)
- TRNCF.DUL: Floating-point convert double to unsigned-long, round toward zero (Double)
- TRNCF.DUW: Floating-point convert double to unsigned-word, round toward zero (Double)
- TRNCF.DW: Floating-point convert double to long, round toward zero (Double)
- TRNCF.SL: Floating-point convert single to long, round toward zero (Single)
- TRNCF.SUL: Floating-point convert single to unsigned-long, round toward zero (Single)
- TRNCF.SUW: Floating-point convert single to unsigned-word, round toward zero (Single)
- TRNCF.SW: Floating-point convert single to long, round toward zero (Single)
- CVTF.HS: Floating-point convert half to single (Single)
- CVTF.SH: Floating-point convert single to half (Single)

(4) Comparison instructions

- CMPF.S: Compare floating-point values (Single)
- CMPF.D: Compare floating-point values (Double)

(5) Conditional move instructions

- CMOVF.S: Floating-point conditional move (Single)
- CMOVF.D: Floating-point conditional move (Double)

(6) Condition bit transfer instruction

• TRFSR: Transfers specified CC bit to Zero flag in PSW (Single)

7.4.3 Conditions for Comparison Instructions

Floating-point comparison instructions (CMPF.D and CMPF.S) perform two floating-point data compare operations. The result is determined based on the comparison condition contained in the data and code. **Table 7.9** lists the mnemonics for conditions that can be specified by comparison instructions.

The comparison instruction result is transferred by the TRFSR instruction to the Z flag of PSW (program status word), and when performing a conditional branch, the condition logic is inverted and then can be used. **Table 7.10** shows logic inversion based on the true/false status of conditions. In a 4-bit condition code for a floating-point comparison instruction, the condition is specified in the "True" column of the table. The conditional branch instruction BT performs a branch when the comparison result is true, while BF performs a branch when the result is false.

| Mnemonic | Definition | Inverte | ed Logic |
|----------|---------------------------------------------------|---------|---------------------------------------|
| F | Always false | (T) | Always true |
| UN | Unordered | (OR) | Ordered |
| EQ | Equal | (NEQ) | Not equal |
| UEQ | Unordered or equal | (OLG) | Ordered and less than or greater than |
| OLT | Ordered and less than | (UGE) | Unordered or greater than or equal to |
| ULT | Unordered or less than | (OGE) | Ordered and greater than or equal to |
| OLE | Ordered and less than or equal to | (UGT) | Unordered or greater than |
| ULE | Unordered or less than or equal to | (OGT) | Ordered and greater than |
| SF | Signaling and false | (ST) | Signaling and true |
| NGLE | Not greater than, not less than, and not equal to | (GLE) | Greater than, less than, or equal to |
| SEQ | Signaling and equal to | (SNE) | Signaling and not equal to |
| NGL | Not greater than and not less than | (GL) | Greater than or less than |
| LT | Less than | (NLT) | Not less than |
| NGE | Not greater than and not equal to | (GE) | Greater than or equal to |
| LE | Less than or equal to | (NLE) | Not less than and not equal to |
| NGT | Not greater than | (GT) | Greater than |



| | | | | Bit Definition of Co | ondition Code fcond(3:0) | | |
|----------|----------------|----------------|-----------|----------------------|--------------------------|---------------------------------------------------------|----------|
| Mnemonic | Conditi fco | on Code ond | Less than | Equal to | Unordered | Invalid operation exception occurs when unordered | Inverted |
| (True) | Decimal | Binary | fcond(2) | fcond(1) | fcond(0) | fcond(3) | (False) |
| F | 0 | 0b0000 | F | F | F | No | (T) |
| UN | 1 | 0b0001 | F | F | Т | No | (OR) |
| EQ | 2 | 0b0010 | F | Т | F | No | (NEQ) |
| UEQ | 3 | 0b0011 | F | Т | Т | No | (OLG) |
| OLT | 4 | 0b0100 | Т | F | F | No | (UGE) |
| ULT | 5 | 0b0101 | Т | F | Т | No | (OGE) |
| OLE | 6 | 0b0110 | Т | т | F | No | (UGT) |
| ULE | 7 | 0b0111 | Т | Т | Т | No | (OGT) |
| SF | 8 | 0b1000 | F | F | F | Yes | (ST) |
| NGLE | 9 | 0b1001 | F | F | Т | Yes | (GLE) |
| SEQ | 10 | 0b1010 | F | Т | F | Yes | (SNE) |
| NGL | 11 | 0b1011 | F | т | Т | Yes | (GL) |
| LT | 12 | 0b1100 | Т | F | F | Yes | (NLT) |
| NGE | 13 | 0b1101 | Т | F | Т | Yes | (GE) |
| LE | 14 | 0b1110 | Т | т | F | Yes | (NLE) |
| NGT | 15 | 0b1111 | Т | Т | Т | Yes | (GT) |

Table 7.10 Definitions of Condition Code Bits and Their Logical Inversions



7.4.4 Floating-Point Instruction Set

This section describes the following items in each instruction (based on alphabetical order of instruction mnemonics).

- Instruction format: Indicates how the instruction is written and its operand(s) (symbols are listed in **Table 7.11**).
- Operation: Indicates the function of the instruction. (symbols are listed in **Table 7.12**).
- Format: Indicates the instruction format (see Section 7.4.1, Instruction formats).
- Opcode: Indicates the instruction opcode in bit fields (symbols are listed in **Table 7.13**).
- Description: Describes the operation of the instruction.
- Supplement: Provides supplementary information on the instruction.

Table 7.11Instruction Format

| Symbol | Explanation |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| reg1 | General-purpose register |
| reg2 | General-purpose register |
| reg3 | General-purpose register |
| reg4 | General-purpose register |
| fcbit | Specifies the bit number of the condition bit that stores the result of a floating- point comparison instruction. |
| imm × | × bit immediate data |
| fcond | Specifies the mnemonic or condition code of the comparison condition of a comparison instruction (for details, see Section 7.4.3, Conditions for Comparison Instructions). |



| Symbol | Explanation |
|--------------|--------------------------------------------------------------|
| \leftarrow | Assignment (input for) |
| GR [a] | Value stored in general-purpose register a |
| SR [a, b] | Value stored in system register (RegID = a , SeIID = b) |
| result | Result is reflected in flag. |
| == | Comparison (true upon a match) |
| + | Add |
| - | Subtract |
| | Bit concatenation |
| × | Multiply |
| ÷ | Divide |
| abs | Absolute value |
| ceil | Rounding in +∞ direction |
| compare | Comparison |
| cvt | Converts type according to rounding mode |
| floor | Rounding in –∞ direction |
| max | Maximum value |
| min | Minimum value |
| neg | Sign inversion |
| round | Rounding to closest value |
| sqrt | Square root |
| trunc | Rounding in zero direction |
| fma(a, b, c) | Result of multiplying a and b and then adding c |
| fms(a, b, c) | Result of multiplying a and b and then subtracting c |

Table 7.12 Operations

Table 7.13 Opcodes

| Symbol | Explanation |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------|
| R | Single bit data of code specifying reg1 |
| r | Single bit data of code specifying reg2 |
| W | Single bit data of code specifying reg3 |
| W | Single bit data of code specifying reg4 |
| I | Single bit data of immediate data (indicates higher bit of immediate data) |
| i | Single bit data of immediate data |
| fff | 3-bit data that specifies the bit number (fcbit) of the condition bit that stores the result of a floating-point comparison instruction |
| FFFF | 4-bit data corresponding to the mnemonic or condition code (fcond) of the comparison condition of a comparison instruction |

| ABSF.D | | | | | | | | | | | | | | | | | F | -loa | ting | -poi | int / | Absol | ute ∖ | /alue | e (D | oubl | e) |
|--------------------------------------|----------|----------------------------------|-----------------------|----------------|---------------------------|--------------|---------------------|----------------------|-------------------------|----------------------|---------------------|-----------------|--------------------|--------------------|---------------------|------------------|------|-------------|--------------|-------|-------------|-----------------|------------|-----------------|--------------|-------------|----|
| | | | | | | | | | | | | | | | Flo | atir | ng-p | ooin | t ab | solu | ute | value | (doi | uble | pre | cisio | n) |
| [Instruction format] | | ABSF.I |) re | g2, r | eg3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | abs | (reg | 2) | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format | F:I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 1 | n | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 2' | 1 20 | 1 | | 17 <i>·</i> | 16 |
| | r r | rr | 0 1 | . 1 | 1 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | w | w | W | w | 0 | 1 | 0 | 0 | 0 | 1 0 | 1 | 1 | 0 | 0 | 0 |
| | <u> </u> | reg2 | | | | | | | | | | | | r | eg3 | | | | cat | ego | ory | type | | sub | -op | | |
| [Description] | | This ins contents register | struc s of pain | tion the ro | takes egiste cified | the er pa | abs air s ger | solu spec nera | ute v cifie al-pu | valu ed b urpo | ie fi y g ose | roi en re | m t iera gis | he d l-p ter | dou urpo reg. | ble ose 3. | -pr | eci gist | sion er 1 | ı flo | oat 2, a | ing-p ind st | oin ore | t foi s it i | rma in tl | t ne | |
| [Floating-point operation exceptions |] | None | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Supplement] | | A subno | orma | al inp | out wi | ill no | ot b | e f | lush | ed | eve | n | if ti | he l | FS ł | oit | oft | the | FP: | SR | reg | gister | is 1 | | | | |



| ABSF.S | | | | | | | | | | | | | | | | | FI | oat | ing-p | oint | t Abso | lute | Valu | 3) al | Sing | le) |
|--------------------------------------|-----|----------------|--------------|------------------|----------------|---------------|------|---------------|---------------|---------------|--------------|--------------|-------------|-----------|--------------|-----------------|--------------|------|-----------------|-------------|------------------|---------------|-------------|------------|-------|--------|
| [Instruction format] | | ABSF. | S 1 | reg2, | reg3 | ; | | | | | | | | | Fl | oatin | g-po | bint | abso | olute | e value | ; (Sir | ngle | pred | cisic |)) |
| [Operation] | | reg3 ← | - al | bs (re | eg2) | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Forma | t F: | I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | | 5 | 4 | | | 0 | 31 | | | 2 | 726 | 6 2 | 5 | 23 | 22 21 | 1 20 | 1 | | 17 | 16 |
| | r r | rr | r | 1 1 | . 1 | 1 | 1 | 1 | 0 0 | 0 | 0 | 0 | w | W | W | wv | 1 | C | 0 | 0 | 1 0 | 0 | 1 | 0 | 0 | 0 |
| | | reg2 | | | | | | | | | | | | r | eg3 | | | с | ateg | ory | type | | sub | o-op | | |
| [Description] | | This in conten | istr ts c | uction of ger | n tak neral | tes t -pui | he a | abso se ro | olute egis | e va ter 1 | ılue reg2 | fro: , ar | m t nd s | the store | sing es i | gle-p t in g | reci gene | sic | on flo I-pui | oati rpo | ing-po se reg | oint ;iste | for r re | mat g3. | | |
| [Floating-point operation exceptions | 5] | None | | | | | | | | | | | | | | | | | | | | | | | | |
| [Supplement] | | A subr | or | mal iı | nput | wil | l no | t be | e flu | she | d ev | ren | if t | he I | FS t | oit o | f th | e F | PSR | re | gister | is 1 | . • | | | |



| ADDF.D | | | | | | | | | | | | | | | | | | | | | Flo | atin | g-po | oint | Ad | d (C |)oub | ole) |
|-----------------------------------------|----|--------------------------------------------------------------------------|----------------------------------------|-------------------------------------|-----------------------------------------|---------------------------------------|----------------------------------|------------------------------------|-----------------------------------|-------------------------------------|-----------------------------|----------------------------------|------------------------------------|----------------------------|-----------------------------------|------------------------------------|-----------------------------|---------------------------------|-------------------------------------|---------------------------------|-----------------------------------|-------------------------------------|--------------------------------------|----------------------------------|-------------------------------------|---------------------------------------|------------------------------------|---------------|
| | | | | | | | | | | | | | | | | | | F | oati | ng- | poiı | nt a | dd (| dou | aldı | pre | cisio | on) |
| [Instruction format] | | ADDF.D | reg | g1, 1 | eg2, | , reg | 3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← r | eg2 | + re | eg1 | | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | :I | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 11 | 10 | | | | F | 4 | | | | 0 | 24 | | | | 27 | 26 | 25 | | 22 | 22 | 01 | 20 | | | 17 | 16 |
| | r | rrr 0 | 1 | 1 | 1 1 | 1 1 | 1 | R | R | R | R | 0 | w | w | w | w | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| | | reg2 | Ì | | | | | | | reg | 1 | | | | reg | 3 | | | ca | tego | ory | ty | ре | | sub | э-ор |) | |
| [Description] | | This instr specified contents the regist were of in mode. | ructi by ; of th er p nfin | on gen ne re air ite a | adds eral- egist spec accu | the -purj er p ifiec racy | do pos air d by , ar | ubl e re spe y ge nd t | e-p egis cifi ener he | orec ster ied ral- rest | reg by g pur ult i | n f 1 v ger pos s re | loa with nera se r our | tinį 1 th 1-p egi | g-p e d ourp ste d ir | oin out oose r re n ac | t fo ble- e re g3. | rma pre gist Th dar | at c cisi ter 1 e o nce | ont on reg2 per wit | ent flo 2, a atic h t | atin atin and an i he o | f th ng-r sto is ex curr | e ro poi res xec ren | egis nt f the cute t ro | ster form e res ed a ounc | pai nat sult s if ding | r in it |
| [Floating-point operation exceptions |] | Unimpler Invalid o Inexact e Overflow Underflo | men pera xcej / exc w ez | ted tion ptio cept xcej | open n exc n (I) tion ptior | ratio cepti (O) n (U | on e ion) | xce (V) | epti) | on | (E) | | | | | | | | | | | | | | | | | |



RH850G3MH Software

[Operation result]

| reg2(B) | | | | | | | | |
|---------|---------|---------|----|----|-----------|-----------|-------|-----------|
| reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
| +Normal | | | | | | | | |
| –Normal | | Δ. | D | | | | | |
| +0 | | A + | Ъ | | | | | |
| -0 | | | | | | | | |
| +∞ | | | | | +∞ | Q-NaN [V] | | |
| -∞ | | | × | | Q-NaN [V] | -∞ | | |
| Q-NaN | | | | | • | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| ADDF.S | | | | | | | | | | | | | | | | | | | | | Flo | oatir | וg-p | oin | t Ado | d (Sin | gle) |
|-----------------------------------------|-----------|---------------------------------------------------------|-----------------------------|---------------------------------------------|----------------------------------------|-------------------------------------|---------------------------------|----------------------------|------------------------------|-------------------------------|------------------------------|----------------------------|-----------------------------|----------------------------|------------------------------|---------------------------|----------------------------|----------------------------|----------------------|-----------------------------|--------------------------|--------------------------|---------------------|---------------------------|-------------------------------|-----------------------------|-----------|
| | | | | | | | | | | | | | | | | | | F | loa | ting | -po | int a | add | (sir | gle p | orecis | ion) |
| [Instruction format] | | ADDF. | S | reg1, | reg | 2, re | eg3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | · re | eg2 + | reg | 1 | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format | F: | I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 4.5 | | | 4.0 | | | _ | | | | | | | | | | ~7 | ~~~ | ~- | | ~~~ | | | | | 4- | |
| | 15 r r | rr | 11 r | 10 1 1 | 1 | 1 | 1 1 | 4 | R | R | R | 0 R | 31 w | w | w | w | 27 w | 26 1 | 25 0 | 0 | <u>23</u> 0 | 1 | 21 1 | 20 0 | 0 | <u> </u> | 0 |
| | | reg2 | | | | | | | | reg | 1 | | | I | reg3 | 3 | | | ca | teg | ory | ty | pe | | sub- | ор | |
| [Description] | | This in: register register as if it roundir | stri re re we | uction g1 w g2, a re of mode | n add ith t nd s infir | ds th he s tore nite | ne si inglo s the accu | ngle e-pr res rac | e-pr ecis sult y, a | recis sior in g nd t | sion n flo gene the | n fl oati era res | oati ing- l-pu ult | ng- poi irpo is r | -poi int i ose rour | int fori reg nde | for mat gist d ii | ma t co er r n ao | t co onte reg3 | onte ents 3. T rda | ents of The nce | s of ger ope wi | ger iera erat | nera al-p ion he | al-pu urpo is e curr | urpos ose xecu ent | se ted |
| [Floating-point operation exceptions |] | Unimpl Invalid Inexact Overflo Underf | len op ex w lov | nente beratio ccepti exce v exc | d op on e on (ption eptio | erat xcer I) n (O on () | ion (ption)) U) | exce (V | epti | on | (E) | | | | | | | | | | | | | | | | |



RH850G3MH Software

[Operation result]

| reg2(B) | | | | | | | | |
|---------|---------|---------|----|----|-----------|------------|-------|-----------|
| reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| +Normal | | | | | | | | |
| –Normal | | Δ. | D | | | ~ | | |
| +0 | | A + | ·В | | | | | |
| -0 | | | | | | | | |
| +∞ | | | | | +∞ | Q-NaN [V] | | |
| -∞ | | | × | | Q-NaN [V] | _∞ | | |
| Q-NaN | | | | | • | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| CEILF.DL | Floating-point Convert Double to Long, round toward positive (Double) |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Conversion to fixed-point format (double precision) |
| [Instruction format] | CEILF.DL reg2, reg3 |
| [Operation] | $reg3 \leftarrow ceil reg2 (double \rightarrow long-word)$ |
| [Format] | Format F:I |
| [Opcode] | |
| | <u>15 11 10 5 4 0 31 27 26 25 23 22 21 20 17 16</u> |
| | r r r r 0 1 1 1 1 1 1 0 0 0 1 0 w w w w 0 1 0 0 0 1 0 1 |
| | reg2 reg3 category type sub-op |
| [Description] | This instruction arithmetically converts the double-precision floating-point format contents of the register pair specified by general-purpose register reg2 to 64-bit fixed-point format, and stores the result in the register pair specified by general-purpose register reg3. The result is rounded in the $+\infty$ direction regardless of the current rounding mode. When the source operand is infinite or not-a-number, or when the rounded result is outside the range of $2^{63} - 1$ to -2^{63} , an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are not enabled, the preservation bit (bit 4) of the FPSR register is set as an invalid operation and no exception occurs. The return value differs as follows, according to differences among sources. |
| | • Source is a negative number, not-a-number, or $-\infty$: -2^{63} is returned. |
| [Floating-point operation exceptions] | Unimplemented operation exception (E) Invalid operation exception (V) Inexact exception (I) |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CEILF.DU | JL | | | | Flc | bating | -pc | oint (| Conv | cc | Do | vers | le to | o Unsi to uns | gneo signe | d-Lo ed fi | ong, xed | round | towa forma | ırd p at (c | bosi dout | tive (ple pr | Doul ecis | ble) ion) |
|-----------------------------------------|------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|---------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------|---------------------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------------------------------------------------------------|----------------------------------------------------------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------|-------------------------------------------------------------------|-------------------------------------------|--------------|
| [Instruction format] | CE | EILF.D | UL 1 | eg2, | reg3 | | | | | | | | | | | | | | | | | | | |
| [Operation] | reg | g3 ← c | eil re | g2 (d | double | $e \rightarrow i$ | un | sign | ied 1 | on | g-v | WO | rd) | | | | | | | | | | | |
| [Format] | Fo | rmat F | :I | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | _ | | | | | - | | | | ~- | | | | | | ~ ~ | | | |
| | 15 | 11 | 10 | | 1 1 | 5 | 4 | 0 | 0 | 1 | 0 | 31 | | | 27 | 26 | 25 | 23 | 3 22 | 21 2 | 20 | 0 1 | | 16 |
| | rrr | r U | L . | | ΙI | 1 | T | U | 0 - | L | 0 | W | W | w w | 0 | | 0 | 0 0 | T | 0 | T | 0 1 | 0 | |
| | reg | 2 | | | | | | | | | | | r | eg3 | | | ca | tegory | typ | е | 5 | sub-o | р | |
| [Description] | Th the for Th Wi res dev If i reg fol | is instr regist mat, a e resul hen the ult is c ected. nvalid gister is lows, a Sour | ructio er pai nd sto t is ro e sour outsid oper s set a accor- rce is | n ari ir spores bund cce o le the atior as an ding a po a ne | thmet ecified the re- ed in perand e rang n exce inval to dif ositive gative | icall d by sult i the + d is i e of f ferer num | y c ge in t $-\infty$ nfi 2^{6^4} ns : per nce nbe | conv enera the r dire inite inite inite are r catio es ar er ou er, r | rerts hl-pu regis ectic e, no l to not e n an mon utsid | the urp ster on 1 ot-a 0, a ena d 1 g s le t | e d oos r pa reg a-nu an abla no cou che um | lou e re air garc um IE ed, exe rat | ble egis spe flles ber EE the cep es. nge | -precision | ision eg2 d by he c ega lefir erva 54^{-} 0 is | n fl ⁱ to u y ge curr tive ned atio urs. 1 to s re | oat: insi ener rent inv on b Th o 0 | ing-po igned al-pu: round umber valid c vit (bit e retu , or + ned. | bint f 64-t rpose ding , or v opera (4) c rn v (4) c rn v (2) | iorn bit f e re mo whe ution of th alue | nat ixe gis ode. en t n e H e di | cont d-po ter ra he ro xcep FPSR ffers s ret | ents int 2g3. ound tion as | led is |
| [Floating-point operation exceptions | Ur] Inv Ine | implei valid og exact e | mente perati xcept | ed op ion e ion (| eratic xcept (I) | on ex ion (| ce V) | ptio) | n (E | 5) | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| CEILF.DU | JW | Flo | pating | j-point | t Conve | ert Do | ouble versio | to Unsig n to uns | gned | -Wo | ord, roun xed-point | d toward | l posit (doub | tive (De | cisic | le) on) |
|-----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|--------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------|-------------------------------|----------------------|
| [Instruction format] | CEILF.DU | W reg2, reg3 | 3 | | | | | | | | | | | | | |
| [Operation] | reg3 ← ceil | l reg2 (double | $e \rightarrow$ | unsig | gned v | vord |) | | | | | | | | | |
| [Format] | Format F:I | | | | | | | | | | | | | | | |
| [Opcode] | | 0 | _ | | | 0 | 04 | | 07 | | 05 0 | 0 00 04 | 00 | | 47 | 10 |
| | 15 11 1 r r r r 0 1 | <u>0</u> 1 1 1 1 1 | 5 1 | 4 | 0 1 | 0 | <u>31</u> w w | 7 547 547 | 27 w | 26 | 25 2 | $\frac{3}{1}$ $\frac{22}{1}$ $\frac{21}{1}$ | 1 | 0 0 | 17 0 | 16 0 |
| | rog2 | | - | ± 0 | 0 1 | | | rog2 | | - | catogor | / ± 0 | | | Ű | |
| [Description] | This instruct the register format, and The result i When the ser result is out detected. If invalid op register is s follows, acc • Source • Source | ction arithmet pair specifie I stores the re is rounded in ource operan tside the rang peration exce tet as an inval cording to dif e is a positive e is a negative | ticall d by sult the - d is ge of eptio lid o ffere e num | ly con genee $+\infty$ di infini 2^{32} – ns are perati- nces nber mber, | ral-pu neral- irectio ite, no - 1 to (e not e ion an amon outsid | the c rrpos purp n reg t-a-n 0, an enabl d no g sou e the i-num | loubl e reg ose r gardle umbe IEE ed, tl exce rces. rces. hber, | e-preci ister re egister ess of the er, or no E754-d he press ption o ge of 2^6 or $-\infty$: | sior g2 t reg2 he c egat efin erv occu 54 _ 0 is | n flo to u 3. currentive ned atio rs. 1 to 5 ref | oating-p nsigned ent rour numbe invalid n bit (bi The retu o 0, or + turned. | oint for 32-bit ading m r, or wh operation t 4) of mn valu ∞ : 2 ³² | mat fixed node. nen tl on ex the F ne dif | conter d-point he rou ception PSR ffers a s return | nts it nde on i s | of ed is d. |
| [Floating-point operation exceptions | Unimpleme] Invalid ope Inexact exc | ented operatio ration except ception (I) | on ex ion (| (V) | ion (E |) | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| CEILF.DV | V | | | | | | | | F | =loa | ting | -po | int (| Cor | ver | rt D | oub | le to | o W | ord, | roun | d to | owa | rd p | osi | itive | (Do | ubl | e) |
|--------------------------------------|-----|----------------------------------------------------|----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------------------------------------|--------------------------------------------------------|------------------------------------------------------------------------|------------------------------------------------------|----------------------------------------------------|-----------------------------------------------------------------|--------------------------------------------|-------------------------------------------------|---------------------------------------------------|-------------------------------------|-------------------------------------------|-------------------------------------|--------------------|----------|
| | | | | | | | | | | | | | | | С | on | vers | ion | to f | xed | -poin | fo | rma | it (de | out | ble p | oreci | sio | n) |
| [Instruction format] | | C | EIL | .F.D | W reg | g2, re | eg3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | re | g3 · | ← c | eil reg | 2 (d | ouble | $e \rightarrow$ | wo | ord) |) | | | | | | | | | | | | | | | | | | |
| [Format] | | Fo | orm | at F | I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | | | 11 | 10 | 1 | 1 1 | 1 | 4 | 0 | 0 | 1 | 0 | 31 | | | | 27 | 26 | 25 | 2 | 32 | <u>22 2</u> | 21 2 | <u>:0</u> 1 | 0 | 1 | 7 ′ | 16 |
| | r . | <u>r</u> 1 | с г | . 0 | 1 1 | T | 1 1 | T | 0 | U | 0 | 1 | 0 | w | W | w | w | w | | 0 | 0 (| | T | 0. | 1 | 0 | <u> </u> | , | 0 |
| | | re | g2 | | | | | | | | | | | | I | reg | 13 | | | ca | tegor | y | type | э | 5 | sub-o | ор | | |
| [Description] | | Th th std Th W ra If re fo | nis i e re ores he r 'her nge inv gist illov | instr egisto s the esul ¹ n the e of 2 alid cer is ws, a Sour Sour | uction er pain result t is rot sourc $2^{31} - 1$ opera set as accord rce is a cce is a | arith spea in g unde to - tion an i ing t neg | hmet cifiec generation 2^{31} , exception invaliant o dif | ical d by al-p the d is an 1 ptic id o ffere nun e nu | ly c y ge y ge y ge y for y for y for y for y for y for y for y for y for y for y for y for y for y for y | conv ner oose dir nite EE7: are catic es a er o er, 1 | vert al-j e rej ect e or 54- not mo r + not | ts tl purj gist ion def t en and ng ∞ : $\frac{1}{2}$ | ne c pos ter : reg t-a- ïne abl no sou 2 ³¹ num | lou e r reg gard-nu ed i led, ex urce | ble egi: 3. dles mb nva , the cep es. 1 is er, c | ste ss per, alic e p otic s re | reci of th or ' l op orese on o turr | sion g^2 the c where g^2 erv erv accumed. -2^3 | n fl to 3 curr en t tion atic urs. | oati 32-t rent he r n ex on b Th | ing-p bit fiz round ccept it (bi it (bi turne | oin di di ior t 4 urn d. | nt fo I-po I res I res I is I oi | orm oint moo sult det f the lue | iat fo de. is ect di | con rma out ted. FPS] ffer | iteni it, a side R s as | ts o nd e tł | of ne |
| [Floating-point operation exceptions |] | U: In In | nim vali | ipler id op ict ez | nenteo peratio ccepti | l ope on ex on (I | eratio cepti | on e ion | xce (V) | ptic | on (| (E) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | -Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CEILF.SL | - | | | | | | | | Flo | atin | ng-p | oir | nt (| Con | ve | rt Sir | ngle | to L | .ong | g, rou | nd | towa | ard | 205 | sitive | e (Sir | ngl | e) |
|--------------------------------------|--------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------------------------------------|------------------------------------------------------------|-------------------------------------------------|----------------------------------------------------------------------------------|-----------------------------------------------------------|---------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|------------------------------------------------------------|----------------------------------------------------------------|---------------------------------|----------------------------------------------------------|-----------------------------------------|------------------------------------|----------------------------------|-----------------------------------|------------|----------------|
| | | | | | | | | | | | | | | | Сс | onve | rsior | n to | fixe | d-poi | nt f | form | at (s | sing | gle p | oreci | sio | n) |
| [Instruction format] | | CEIL | .F.SI | L reg2 | l, reg | 3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ← c | eil reg | 2 (sir | ngle – | →] | lon | g-wo | ord | l) | | | | | | | | | | | | | | | | | |
| [Format] | | Form | at F | :I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 14 | 10 | | | F | 4 | | | | 0 | 2 | 4 | | | 07 | | 05 | | 5 | <u> </u> | 1 0 | 0 | | 1. | 7 4 | 16 |
| | r r | rrı | - 11 : r | 1 1 | 1 1 | L 1 | 5 1 | 4 | 0 | 0 | 1 | 0 | 3 1 | n v w | , , | w w | 27 | 1 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 0 | | 0 |
| | | reg2 | | | | | | | | | | | | | re | eg3 | | | са | itegor | у | type | e | ę | sub- | ор | |] |
| [Description] | | This is gener pair s The r When range If inv regist follow | instr al-p peci esul n the e of 2 alid cer is ws, a Sour | ruction urpose ified b t is rou- source $2^{63} - 1$ opera s set as accord rce is a rce is a | arith e regi y gen undec e ope to -2 tion e an in ing to a posi | ster ru eral-j l in th rand ¹ 2 ⁶³ , a excep nvalic o diffe tive r | cal eg2 pur is i is i fio 1 o ere nur | ly of 2 to rpo +∞ infi IEE ns per ence mbe | conv o 64- se re dire nite EE75 are r ration es an er or er, n | ert bit egis ccti or 4-c not not +o | ts th fix ster on not defi ena nd ng s co: 2 -a-n | ne ced reg ;-a- ine abl no sou 263 | sin l-p ga -n ed leo e uro | ngle poin 3. rdle uml inv 1, th xce ces. 1 is per, | e-p nt f ess be ali ne pti s r or | form form a of t r, or id op pres ion of etur $-\infty$ | sion at, the own white the serves becomes a serves of the | n fl and curi en itio atio irs. | oati l sto rent the : n ex on b Th Th | ing-p pres t t roun roun kcept bit (b be ret | oin he ion it 4 urn | nt fo rest ing p d res n is 1) of n va | orm ult sult det the lue | at in de. is ect di | con the out ted. FPS | tent regi side R s as | s c ste | of er ne |
| [Floating-point operation exceptions |] | Unim Inval Inexa | ipler id oj ict ez | nenteo peratio xceptio | l oper on exc on (I) | ration | n ez on (| xce (V) | ptio | n (1 | E) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|-------------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (integer) | | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CEILF.SU | JL | | | | | F | loat | ing | -poir | nt C | Conv | /ert Cor | Si nv | ingle ersie | e t on | o Un to u | signe nsigr | ed-l | Lon fixe | g, r ed- | rounc point | for | war mat | d po t (si | osi ingl | tive le pi | (Sin | igle) sion) |
|--------------------------------------|---------|------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------|------------------------------------------------------------------|-------------------------------------------------------------------|-----------------------------------------------------------------|--------------------------------------------------------------|----------------------------------------------------------|------------------------------------------------------------------------|---------------------------------------------------|-----------------------------------------------------------|--------------------------------------------------------------|------------------------------------------|----------------------------------------------------------|---------------------------------|---------------------------------------------------------|-------------------------------------------------------|--------------------------------------------------|----------------------------------------------------|------------------------------------------------|---------------------------------------------------------|-----------------------------------------|------------------------------------------|------------------------------------------|-------------------------------------|-------------------------------------|----------------------|----------------|
| [Instruction format] | | CEILF.S | UL | reg | 2, re | g3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | $reg3 \leftarrow c$ | ceil 1 | reg2 | 2 (sir | ıgle | \rightarrow | uns | sign | ned | loı | ng-v | wo | ord) |) | | | | | | | | | | | | | |
| [Format] | | Format F | ₹:I | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | | | | _ | | | | | 0 | 0 | | | | 0- | | | - | 00 | 00 | | | - | | 4- | |
| | 15 r | 11 | 10 | 1 | 1 1 | 1 | 1 | 4 | 0 | 0 | 1 | 0 | 3 | 51 | .7 | 747 7 | 21 | 1 | | | 23 | 1 | 21 | |) | | 1/ | 16 |
| | | | | | 1 1 | | 1 | | 0 | 0 | T | 0 | | ~ ~ | v | | v 0 | | | | 5 0 | | | | | , <u> </u> | . 0 | |
| | | reg2 | | | | | | l | | | | | ļ | | r | eg3 | | | C | ate | gory | ty | pe | | รเ | o-di | р | |
| | | specified the result The result When the result is of detected. If invalid register i follows, | by t in t lt is e sou outsi l ope s set accc | gene the r rour urce ide t erati t as ordin | eral- regis nded ope the r ion e an ir ng to | purj ter j l in t rang ang exce tval o dif | pose pair the d is e of ptio id o ffere | e re sp $+\infty$ inf 2^{6} ons open | egis ecif dir init ⁱ⁴ – are ratio es a | ter fiec rect e, 1 1 t no on | reg l by tior not co 0 ot er anc ong | g2 tr 7 ge 1 reg -a-r , an nab l nc sou | o ga nu n I ele o e ur | uns eral ardl mb EE d, ti exce ces. | ig -f es er E he | ned ourpo s of , or 1 754- e pre tion | 64-l ose i the nega defin serv occi | bit reg cur ativ nec rati- urs | fixe iste ren e n l in on 1 . Tl | ed- er r nu r num val bit he | poir eg3. ound hber, lid o (bit retur | nt fo ling or per 4) m v | orm g m wh atio of a valu | nat, node nen on the ue c | , an e. th ex FI dif | nd s le ro cep PSR fers | tore ound tion | ded is |
| | | • Sou | rce i | is a | posi | tive | nui | nb | er o | outs | side | the | e 1 | rang | ge | of 2 | | - 1 | to (|), с | or +o | 0:2 | , | - 1 | 15 | , ret | urn | ed. |
| | | • Sou | rce i | is a : | nega | ative | e nu | mb | ber, | not | t-a- | nur | mł | ber, | 0 | r –∞ | : 0 i | s r | etu | rne | ed. | | | | | | | |
| [Floating-point operation exceptions | 5] | Unimple Invalid o Inexact e | men pera xcej | ited itior ptio | open 1 exc n (I) | ratic cepti | on e: ion | xce (V | eptio) | on | (E) | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

reg2 (A) +Normal -Normal +0

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|------------|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.
| CEILF.SU | JW | | | | FI | loatii | ng-ı | poin | t Co | onve C | rt S | ver | le t | o Ur n to u | nsigne unsig | ed-\ | Nor I fixe | d, ro ed-po | und pint | forn | rard nat (| pos (sing | sitive gle p | e (Sin | igle |) |
|-----------------------------------------|--------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|-----------------------------|-------------------------------------------------------------------|---------------------------------------------------------------------|--------------------------------------------------------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|----------------------------------------------------------------------|---------------------------------------------------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------------------------------------|------------------------------------------------------|-----------------------------------------------|----------------------------------------------|------------------------------------------------------|--------------------|---|
| [Instruction format] | CI | EILF.SU | JW r | eg2, | reg3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | reş | g3 ← c | eil reg | g2 (si | ngle | →ı | uns | signo | ed v | wor | d) | | | | | | | | | | | | | | | |
| [Format] | Fo | rmat F | :I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | _ | | | | | _ | | | | | | | _ | | | | | | . – | | - |
| | 15 | 11 | 10 | 1 | 1 1 | 1 | 4 | 0 | 0 | 1 | 0 | <u>31</u> | | | 27 | 7 2 | <u>5 25</u> | 5 | 23 | 22 | 21 2 | 20 | 0 | 17 | | 3 |
| | rrr | r r | | Ţ | ΙI | 1 | | 0 | U | 1 | 0 | w | W | w | w w | | . 0 | 0 | U | 1 | 0 | 0 | | 0 0 | | |
| [Description] | Th the for Th W res de If reg fol | is instr e register mat, ar e resul hen the sult is o tected. invalid gister is lows, a Sour Sour | uction er pain nd sto t is ro source outside operations set a accord rce is a rce is a | n arit r spectres th unde ce op e the ation s an i ling t a pos | hmet cified ne res d in t erand range excep invali o diffi itive gative | ical d by sult the - d is e of ptio id o fere nur | ly c ge in c $+\infty$ inf 2^{32} ms per ence mbe | conv enera gene dire inite ² – 1 are ratio es an er on er, 1 | vert al-p eral ecti- e, no 1 to not mor utsi- not- | ena nd 1 ng s de t | ale s rpo reg a-n an abl no cou the um | sing e re ose garc um IE ed, rce rce ran | gle- egis reg ber EE the cep es. nge | pre- ster giste s of ; or 754 e pro- tion = of : r $-\infty$ | cisio reg2 er reg f the nega -defi eserv cocc 2^{64} - | n f to g3. cur ativ nec vati urs - 1 is r | loat uns rren e n l in on l . Th to () etui | ing- igne t rou umb valid oit (l ne re), or rned | poi ed 3° und ber, d op bit etur $+\infty$ | int 1 32-t ling or 1 pera 4) c rn v | iorn bit f mo whe ation of th alue | nat fixe ode en t n e He di | con d-po he r xcep FPSI ffers | tents pint ound obtion R s as turn | s o dec i is | E |
| [Floating-point operation exceptions | Ur 5] Inv Ine | nimpler valid op exact ex | nente peratio xcepti | d ope on ex on (I | eratio cepti | on ex ion (| xce (V) | ptio) | on (1 | E) | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| CEILF.SV | V | | | | | | | | Flo | atir | ng-p | oin | t Co | onve | ert S | ingle | to | Woi | d, ro | ound | tow | ard | pos | sitive | (Sin | gle) |
|------------------------------------------|----|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|------------------------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------------------------------|-------------------------------------------------------------------------|-------------------------------------------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------|--------------------------------------------------------------------------|----------------------------------------|---------------------------------------------------------|--------------------------------------------------|----------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------|-----------------------------------------------------|-----------------------------|------|
| | | | | | | | | | | | | | | C | Conv | ersic | n to | o fix | ed-p | point | form | at (| sing | gle pr | ecisi | on) |
| [Instruction format] | | CEI | LF.S | W reg | g2, re | eg3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ¦ ← c | eil reg | g2 (si | ingle | \rightarrow | wo | ord) | | | | | | | | | | | | | | | | | |
| [Format] | | For | nat F | :I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | | 11 | 10 | 1 | 1 1 | 5 | 4 | 0 | 0 | 1 | 0 | 31 | | | 2 | 72 | 62 | 5 | 23 | 22 | 21 2 | 20 | | 17 | 16 |
| | r | r r | r r | | Ţ | ΙI | T | 0 | 0 | 0 | T | 0 | W | W | W | w v | 7 _ | | 0 0 | 0 | T | 0 | 0 | 0 0 | 0 | 0 |
| | | reg2 | | | | | | | | | | | | I | eg3 | | | С | ateg | gory | typ | е | 5 | sub-o | р | |
| [Description] | | This genu purp The Who rang If in regi follo | s insta eral-p pose i resul en the ge of valid ster i sows, s Sou | ruction purpos registe It is ro 2^{31} – l opera s set a accord rce is rce is | n arit e reg r reg unde ce op 1 to - tion s an ling t ling t a pos | hmet (3. ed in t erand -2^{31} , exceptions to diffustive gative | ical reg: the l is i an l ptio id o fere nur e nu | ly 2 to +∞ inf EF nc nc nc nb mb | conv o 32 inite EE7: are ratic es ar er or | vert -bit ecti c or 54-c not on a motion r +c not- | ts th t fix ion not defi i ena ind ng s ∞ : 2 -a-n | reg t-a- ine abl no sou 2 ³¹ | sing -pc garce num d in ed, exe urce - 1 nbe | gle- bint dles mb nva the cep es. l is er, c | for for ss of er, c lid c e pro- tion retu | cisic mat, f the or wh oper eserv cocc | on f an cu nen atio vati turs 1. 2 ³¹ | loa d st the on e on s. T. | ting fore nt rot e rot excee bit he r | g-poi ound ounde eptio (bit retur | nt f e res ing d re n is 4) o n v e | orm ault mo sul de ¹ f th alue | nat in ode. t is tec ne F e di | cont gene outs ted. PSF ffers | ents ral- ide t as | of |
| [Floating-point operation exceptions] |] | Uni Inva Inex | mple ilid o act e | mente perati xcepti | d ope on ex on (l | eratio ccepti [) | on ex ion | xce (V) | eptic) | on (| E) | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



Floating-point Conditional Move (Double) CMOVF.D Conditional move (double precision) [Instruction format] CMOVF.D fcbit, reg1, reg2, reg3 [Operation] if (FPSR.CCn == 1) then $reg3 \leftarrow reg1$ else $reg3 \leftarrow reg2$ endif Note 1. n = fcbit [Format] Format F:I [Opcode] 5 4 0 31 27 26 25 23 22 21 20 15 11 10 17 16 0 0 0 0 0 1 f f rrrr 0 1 1 1 1 1 1 R R R R O w w w w O 1 f 0 reg3*1 reg2 category type sub-op reg1 Caution 1. reg3: wwww! = 0 wwww \neq 0000 (do not set reg3 to r0) Note: fcbit: fff [Description] When the CC(7:0) bits of the FPSR register specified by fcbit in the opcode are true (1), data from the register pair specified by reg1 is stored in the register pair specified by reg3. When these bits are false (0), data from the register pair specified by reg2 is stored in the register pair specified by reg3. [Floating-point None operation exceptions] [Supplement] A subnormal input will not be flushed even if the FS bit of the FPSR register is 1. CAUTION Do not set reg3 to r0.



<Floating-point condition instruction >

| CMOVF.S | 5 | | | | | | | | | Floa | ing-p | oint C | onditio | nal N | love (S | Single) |
|------------------------------------------|-------------------------------------------|-------------------------------------------------------|---------------------|----------------|-----------|-----------|------------|-----------------|----------------|---------------|----------------|-----------------|--------------------|-----------------|------------------|---------|
| | | | | | | | | | | | Cond | itiona | l move | (sing | gle preo | cision) |
| [Instruction format] | CMOVF. | S fcbit, reg1 | , reg2, | reg3 | | | | | | | | | | | | |
| [Operation] | if (FPSR reg3 else reg3 endif | CCn == 1) t $\leftarrow reg1$ $\leftarrow reg2$ | hen | | | | | | | | | | | | | |
| | Note | 1. n = fcbit | | | | | | | | | | | | | | |
| [Format] | Format F | :I | | | | | | | | | | | | | | |
| [Opcode] | 15 14 | 10 | F | 4 | | 0 | 04 | | | 7.00 | - 05 | 00 | 22.24 | 20 | | 17 10 |
| | rrrr | 1 1 1 1 | 1 1 | R R | R R | R | w | w w | w | v 1 | 0 | 0 0 | 0 0 | 0 | f f | f 0 |
| | reg2 | | | | reg1 | | | reg3 | *1 | | cate | egory | type | s | sub-op | |
| | Caution 1. | reg3: wwwww wwwww ≠ 000 it: fff | r! = 0 000 (dc | not se | t reg3 to | o r0 |)) | | | • | | | | | | • |
| [Description] | When the from reg1 | CC(7:0) bits is stored in | s of the reg3. V | e FPSI When | R regist | er its | spe are | cifiec false | 1 by f (0), | cbit the 1 | in th eg2 o | e opo data i | code a is store | re tri ed in | ue (1), reg3. | data |
| [Floating-point operation exceptions] | None] | | | | | | | | | | | | | | | |
| [Supplement] | A subnor | mal input wi | ll not b | e flus | hed eve | en | if tł | ne FS | bit o | f the | e FPS | R re | gister | is 1. | | |
| | CAUTION | ١ | | | | | | | | | | | | | | |
| | Do not se | t reg3 to r0. | | | | | | | | | | | | | | |



CMPF.D

Compare floating-point values (Double)

loating-point comparison (double precision)

| [Instruction format] | CMPF.D fcond, reg2, reg1, fcbit |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | CMPF.D fcond, reg2, reg1 |
| [Operation] | <pre>if isNaN(reg1) or isNaN(reg2) then result.less ← 0 result.equal ← 0 result.unordered ← 1 if fcond[3] == 1 then Invalid operation exception is detected. endif</pre> |
| | else result.less ← reg2 < reg1 result.equal ← reg2 == reg1 result.unordered ← 0 |
| | endif |
| | FPSR.CCn ← (fcond[2] & result.less) (fcond[1] & result.equal) (fcond[0] & result.unordered) |
| | Note 1. n: fcbit |
| [Format] | Format F:I |
| [Opcode] | |
| | <u>15 11 10 5 4 0 31 27 26 25 23 22 21 20 17 16</u> |
| | r r r r 0 1 1 1 1 1 1 R R R R 0 0 F F F F 1 0 0 0 0 1 1 f f f 0 |
| | reg2 reg1 category type sub-op |
| | Note: fcond: FFFF fcbit: fff |

[Description]

This instruction compares the double-precision floating-point format contents of the register pair specified by general-purpose register reg2 with the double-precision floating-point format contents of the register pair specified by general-purpose register reg1, based on the condition "fcond", and sets the result (1 if true, 0 if false) to the condition bits (the CC(7:0) bits: bits 31 to 24) in the FPSR register specified by fcbit in the opcode. If fcbit is omitted, the result is set to the CC0 bit (bit 24).

For description of the comparison condition "fcond" code, see **Table 7.14, Comparison Conditions**.

If one of the values is not-a-number, and the MSB of the comparison condition "fcond" has been set, an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are enabled, the comparison result is not set and processing is passed to the exception.

If the enable bits are not set, no exception occurs, and the preservation bit (bit 4) of the FPSR register is set, then the comparison result is set to the CC(7:0) bits of the FPSR register. When SignalingNaN (S-NaN) is acknowledged as an operand value in a floating-point instruction (including a comparison), it is regarded as an invalid operation condition. When using only S-NaN but also QuietNaN (Q-NaN) for a comparison that is an invalid operation, it is simpler to use a program in which any NaN results in an error. In other words, there is no need to insert code that checks for Q-NaN that would result in an unordered result. Instead, the exception handling system should perform error processing when an exception occurs after detecting an invalid operation. The following shows a comparison that checks for equivalence of two numerical values and triggers an error when an unordered result is detected.

| Comparis Condition | son ns | Definition | Description | Detection of invalid operation exception by unordered |
|-----------------------|-----------|---------------|-----------------------------------------------------------------------------|----------------------------------------------------------------------|
| F | 0 | FALSE | Always false | No |
| UN | 1 | Unordered | One of reg1 and reg2 is not-a-number | No |
| EQ | 2 | reg2 = reg1 | Ordered (both reg1 and reg2 is not not-a-number) and equal | No |
| UEQ | 3 | reg2 ? = reg1 | Unordered (at least, one of reg1 and reg2 is not-a-number) or equal | No |
| OLT | 4 | reg2 < reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than | No |
| ULT | 5 | reg2 ? < reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than or equal to | No |
| OLE | 6 | reg2 ≤ reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than or equal to | No |
| ULE | 7 | reg2 ? ≤ reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than or equal to | No |
| SF | 8 | FALSE | Always false | Yes |
| NGLE | 9 | Unordered | One of reg1 and reg2 is not-a-number | Yes |
| SEQ | 10 | reg2 = reg1 | Ordered (both reg1 and reg2 are not not-a-number) and equal | Yes |
| NGL | 11 | reg2 ? = reg1 | Unordered (one of reg1 and reg2 is not-a-number) or equal | Yes |
| LT | 12 | reg2 < reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than | Yes |
| NGE | 13 | reg2 ? < reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than | Yes |
| LE | 14 | reg2 ≤ reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than or equal to | Yes |
| NGT | 15 | reg2 ? ≤ reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than or equal to | Yes |

Table 7.14 Comparison Conditions

Note: ?: Unordered (invalid comparison)



```
# When explicitly testing Q-NaN
                              CMPF.D
                                            OLT, r12, r14, 0 # Check if r12 < r14
                                            UN, r12, r14, 1 # Check if unordered
                              CMPF.D
                             TRFSR
                                            0
                             BT
                                            L2
                                                                  # If true, go to L2
                             TRFSR
                                            1
                             ΒT
                                            ERROR
                                                                  # If true, go to error processing
                       # Enter code for processing when neither unordered nor r12 < r14
                             L2:
                       # Enter code for processing when r12 < r14
                              ...
                       # When using a comparison to detect Q-NaN
                                            LT, r12, r14, 0
                              CMPF.D
                                                                  # Check if r12 ?< r14
                             TRFSR
                                            0
                             ВT
                                            L2
                                                                  # If true, go to L2
                       # Enter code for processing when not r12 < r14
                             L2:
                       # Enter code for processing when r12 < r14
                              ...
[Floating-point
                       Invalid operation exception (V)
operation exceptions]
[Supplement]
                       A subnormal input will not be flushed even if the FS bit of the FPSR register is 1.
```



Section 7 INSTRUCTION

RH850G3MH Software

[Operation result]

[Condition code (fcond) = 0 to 7]

| reg1(B) reg2(A) | +Normal | -Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN | | | | |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------|---------|----|----|----|----|-------|-------|--|--|--|--|
| ±Normal | _ | | | | | | | | | | | |
| ±0 | Stores result of comparison (true or false) executed under the comparison condition (fcond) in the FPSR.CCn bit (n = fcbit) | | | | | | | | | | | |
| ±∞ | comparison condition (fcond) in the FPSR.CCn bit (n = fcbit) | | | | | | | | | | | |
| Q-NaN | Unordered | | | | | | | | | | | |
| S-NaN | Unordered [V] | | | | | | | | | | | |

[Condition code (fcond) = 8 to 15]

| reg1(B) reg2(A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN | | | | | |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------|---------|-----|--------|-----------|----|-------|-------|--|--|--|--|--|
| ±Normal | _ | | | | | | | | | | | | |
| ±0 | Stores result of comparison (true or false) executed under the comparison condition (fcond) in the FPSR.CCn bit ($n = fcbit$) | | | | | | | | | | | | |
| ±∞ | | | (, | | | | | | | | | | |
| Q-NaN | | | | Upord | ared [\/] | | | | | | | | |
| S-NaN | | | | UNDIDE | | | | | | | | | |

Note: [] indicates an exception that must occur.



CMPF.S

Compare floating-point values (Single)

Floating-point comparison (single precision)

| [Instruction format] | CMPF.S fcond, reg2, reg1, fcbit CMPF.S fcond, reg2, reg1 |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Operation] | <pre>if isNaN(reg1) or isNaN(reg2) then result.less ← 0 result.equal ← 0 result.unordered ← 1 if fcond[3] == 1 then Invalid operation exception is detected. endif</pre> |
| | else result.less \leftarrow reg2 < reg1 result.equal \leftarrow reg2 == reg1 result.unordered $\leftarrow 0$ |
| | endif |
| | FPSR.CCn ← (fcond[2] & result.less) (fcond[1] & result.equal) (fcond[0] & result.unordered) |
| | Note 1. n: fcbit |

[Format] Format F:I

[Opcode]

| 15 | | | | 11 | 10 | | | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 | 21 | 20 | | | 17 | 16 |
|----|---|------|------|----|-------------|------|-----|---|---|---|---|---|------|---|---|----|---|---|---|----|----|----|-----|-----|----|----|----|-----|------|----|-----------------|
| r | r | r | r | r | 1 | 1 | 1 | 1 | 1 | 1 | R | R | R | R | R | 0 | F | F | F | F | 1 | 0 | 0 | 0 | 0 | 1 | 0 | f | f | f | 0 |
| | I | reg2 | 2 | | | | | | | | | I | reg′ | 1 | | | | | | | | ca | teg | ory | ty | ре | | sub | o-op | | $\overline{ }$ |
| | I | Note | ə 1. | f | con cbit | d: E | FFF | F | | | | | | | | | | | | | | | | | | | | | | | |

[Description]

This instruction compares the single-precision floating-point format contents of generalpurpose register reg2 with the single-precision floating-point format contents of generalpurpose register reg1, based on the comparison condition "fcond", then sets the result (1 if true, 0 if false) to the condition bits (the CC(7:0) bits: bits 31 to 24) in the FPSR register specified by fcbit in the opcode. If fcbit is omitted, the result is set to the CC0 bit (bit 24). For description of the comparison condition "fcond" code, see **Table 7.15, Comparison Conditions**.

If one of the values is not-a-number, and the MSB of the comparison condition "fcond" has been set, an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are enabled, the comparison result is not set and processing is passed to the exception.

If the enable bits are not set, no exception occurs, and the preservation bit (bit 4) of the FPSR register is set, then the comparison result is set to the CC(7:0) bits of the FPSR register. When SignalingNaN (S-NaN) is acknowledged as an operand value in a floating-point instruction (including a comparison), it is regarded as an invalid operation condition. When using only S-NaN but also QuietNaN (Q-NaN) for a comparison that is an invalid operation, it is simpler to use a program in which any NaN results in an error. In other words, there is no need to insert code that explicitly checks for Q-NaN that would result in an unordered result. Instead, the exception handling system should perform error processing when an exception occurs after detecting an invalid operation. The following shows a comparison that checks for equivalence of two numerical values and triggers an error when an unordered result is detected.

| Compariso Conditions | n | | | Detection of invalid |
|-------------------------|-------|---------------|-----------------------------------------------------------------------------|---------------------------|
| | fcond | Definition | Description | exception by unordered |
| F | 0 | FALSE | Always false | No |
| UN | 1 | Unordered | One of reg1 and reg2 is not-a-number | No |
| EQ | 2 | reg2 = reg1 | Ordered (both reg1 and reg2 is not not-a-number) and equal | No |
| UEQ | 3 | reg2 ? = reg1 | Unordered (at least, one of reg1 and reg2 is not-a-number) or equal | No |
| OLT | 4 | reg2 < reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than | No |
| ULT | 5 | reg2 ? < reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than or equal to | No |
| OLE | 6 | reg2 ≤ reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than or equal to | No |
| ULE | 7 | reg2 ? ≤ reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than or equal to | No |
| SF | 8 | FALSE | Always false | Yes |
| NGLE | 9 | Unordered | One of reg1 and reg2 is not-a-number | Yes |
| SEQ | 10 | reg2 = reg1 | Ordered (both reg1 and reg2 are not not-a-number) and equal | Yes |
| NGL | 11 | reg2 ? = reg1 | Unordered (one of reg1 and reg2 is not-a-number) or equal | Yes |
| LT | 12 | reg2 < reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than | Yes |
| NGE | 13 | reg2 ? < reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than | Yes |
| LE | 14 | reg2 ≤ reg1 | Ordered (both reg1 and reg2 are not not-a-number) and less than or equal to | Yes |
| NGT | 15 | reg2 ? ≤ reg1 | Unordered (one of reg1 and reg2 is not-a-number) or less than or equal to | Yes |

| Table 7.15 | Comparison | Conditions |
|------------|------------|------------|
|------------|------------|------------|

Note: ?: Unordered (invalid comparison)

When explicitly testing Q-NaN OLT, r12, r13, 0 CMPF.S # Check if r12 < r14 UN, r12, r13, 1 CMPF.S # Check if unordered TRFSR 0 BTL2 # If true, go to L2 TRFSR 1 BTERROR # If true, go to error processing # Enter code for processing when neither unordered nor r12 < r14 L2: # Enter code for processing when r12 < r14 ... # When using a comparison to detect Q-NaN LT, r12, r13, 0 # Check if r12 ?< r14 CMPF.S TRFSR 0 ΒT L2 # If true, go to L2 # Enter code for processing when not r12 < r14L2: # Enter code for processing when r12 < r14 ••• [Floating-point Invalid operation exception (V) operation exceptions] [Supplement] A subnormal input will not be flushed even if the FS bit of the FPSR register is 1.



RH850G3MH Software

[Operation result]

[Condition code (fcond) = 0 to 7]

| reg1(B) reg2(A) | +Normal | ormal -Normal +0 -0 +∞ -∞ Stores result of comparison (true or false) executed under the comparison condition (fcond) in the FPSR.CCn bit (n = fcbit) Unordered | | Q-NaN | S-NaN | |
|--------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------|------------------|-------|--|
| ±Normal | _ | | | | | |
| ±0 | Stor the cor | res result of (mparison cor | comparison (idition (fcond | nder = fcbit) | | |
| ±∞ | | | | | | |
| Q-NaN | | | | , | | |
| S-NaN | | | | | | |

[Condition code (fcond) = 8 to 15]

| reg1(B) | | | | | | | | |
|---------|-----------------|----------------------------------|-------------------------------|------------------|-------|-------|--|--|
| reg2(A) | +Normal | –Normal | +0 | _∞ | Q-NaN | S-NaN | | |
| ±Normal | _ | | | | | | | |
| ±0 | Stor the cor | res result of on moarison cor | comparison (dition (fcond | nder = fcbit) | | | | |
| ±∞ | | | | | | | | |
| Q-NaN | | | | | | | | |
| S-NaN | | | | | | | | |

Note: [] indicates an exception that must occur.



| CVTF.DL | | | | | | | | | | | | | | F | Float | ing- | poi | nt C | onve | rt [| Dout | ole t | οL | .ong | (Do | oub | le) |
|------------------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-------|---------------|-----|------|-----|----|---|---|----------------|-----|-------|------|------|------|--------|-------|------|-------|------|------|------|-------|-----|
| | | | | | | | | | | | | | C | Cor | vers | ion | to f | ixec | d-poir | nt fo | orma | at (d | loul | ble | ored | cisic | on) |
| [Instruction format] | | CVTF.DI | 2 reg2 | l, reş | g3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← c | vt reg | 2 (do | ouble | \rightarrow | lor | ıg-v | vor | d) | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | :I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 44 | 10 | | | F | 4 | | | | 0 | 2 | 1 | | | 07 | 26 | . OE | . , | 22 | 22 | 04 | 20 | | | 47 | 16 |
| | r r | rrr 0 | 1 1 | 1 | 1 1 | 5 1 | 4 | 0 | 1 | 0 | 0 | 3 | v w | v | v w | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| | | reg2 | r r 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 <td></td> | | | | | | | | | | | | | | | | | | | | | | | | |
| [Description] | | reg2reg3categorytypesub-opThis instruction arithmetically converts the double-precision floating-point format contents of the register pair specified by general-purpose register reg2 to 64-bit fixed-point format, in accordance with the current rounding mode, and stores the result in the register pair specified by general-purpose register reg3.When the source operand is infinite or not-a-number, or when the rounded result is outside the range of $2^{63} - 1$ to -2^{63} , an IEEE754-defined invalid operation exception is detected.If invalid operation exceptions are not enabled, the preservation bit (bit 4) of the FPSR register is set as an invalid operation and no exception occurs. The return value differs as follows, according to differences among sources.•Source is a positive number or $+\infty$: $2^{63} - 1$ is returned.•Source is a negative number, not-a-number, or $-\infty$: -2^{63} is returned. | | | | | | | | | | | of ed he | | | | | | | | | | | | | | |
| [Floating-point operation exceptions] |] | Unimplemented operation exception (E) Invalid operation exception (V) Inexact exception (I) | | | | | | | | | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | al -Normal +0 -0 (integer) 0 (integer) | | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|---------|--------------------------------------------------------------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CVTF.DS | | | | | | | | | | | 1 | -loai | ing-p | oint | Conv | /ert L |)ouble 1 | | igle (| Dout | ole) |
|------------------------------------------|-----------|------------------------------------------------------------|--------------------------------------------------|--------------------------------------------------|--------------------------------------|----------------------------|---------------------------------|--------------------------------|---------------------------|---------------------|---------------------|--------------------------|------------------------|-----------------------|-------------------------|----------------|---------------------------------|----------------------|----------------------|----------------|----------|
| | | | | | | | | | | C | Jony | /ersi | on to | tioa | ting-p | point | format | (aour | ne bi | recisi | on) |
| [Instruction format] | | CVTF.D | S reg2 | , reg3 | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← c | vt reg2 | 2 (doub | $le \rightarrow$ | sin | gle) | | | | | | | | | | | | | | |
| [Format] | | Format F | :I | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | 10 | | r | 4 | | | 0 | 24 | | | 07 | | 05 | 00 | 00.04 | 00 | | 47 | 40 |
| | 15 r r | rr 0 | 1 1 | 1 1 | 5 1 1 | 0 | 0 0 | 1 | 1 | 31 w | W | W | 21 w w | 1 | 0 | 0 0 | 1 0 | 1 | 0 0 |) 1 | 0 |
| | | reg2 | | | | | | | | | r | eg3 | | | cate | gory | type | 5 | ub-o | р | |
| [Description] | | This instr the regist format, a accordan | ruction er pair nd stor ce with | arithm specifi res the r n the cu | etical ied by result irrent | ly c ger in g rou | onve neral gener nding | rts ti -pur al-p g mo | he o pos urp ode | lou se re ose | ble- egis reg | -pre- iter i giste | cisio reg2 r reg | n fl to s ;3.] | oatin ingle The r | e-pre esult | oint for ecision t is rou | rmat floa inde | cont ting 1 in | tents -poir | of nt |
| [Floating-point operation exceptions] |] | Unimpler Invalid o Inexact e Overflow Underflo | mented peratio xceptic excep w excep | l operat on exception (I) otion (Coption (| tion e ption)) U) | xcej (V) | ption | (E) | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|----|----|----|----|-------|--------------|
| Operation result [exception] | A (Si | ngle) | +0 | -0 | +∞ | _∞ | Q-NaN | Q-NaN [V] |

Note: [] indicates an exception that must occur.



| CVTF.DU | L | Floating-point Convert Double to Unsigned-Long (Double) Conversion to unsigned fixed-point format (double precision) | | | | | | | | | | |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|--|--|--|--|--|--|--|--|--|--|
| [Instruction format] | CVTF.DUL reg2, reg3 | | | | | | | | | | | |
| [Operation] | $reg3 \leftarrow cvt reg2 (double \rightarrow ur)$ | nsigned long-word) | | | | | | | | | | |
| [Format] | Format F:I | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | |
| | 15 11 10 5 4 | 4 0 31 27 26 25 23 22 21 20 17 16 | | | | | | | | | | |
| | r r r r 0 1 1 1 1 1 1 | 1 0 1 0 0 w w w w 0 1 0 0 0 1 0 1 0 1 0 | | | | | | | | | | |
| | reg2 | reg3 category type sub-op | | | | | | | | | | |
| [Description] | This instruction arithmetically converts the double-precision floating-point format contents of the register pair specified by general-purpose register reg2 to unsigned 64-bit fixed-point format, in accordance with the current rounding mode, and stores the result in the register pair specified by general-purpose register reg3. When the source operand is infinite, not-a-number, or negative number, or when the rounded result is outside the range of 2⁶⁴ – 1 to 0, an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are not enabled, the preservation bit (bit 4) of the FPSR register is set as an invalid operation and no exception occurs. The return value differs as follows, according to differences among sources. Source is a positive number outside the range of 2⁶⁴ – 1 to 0, or +∞: 2⁶⁴ – 1 is returned. Source is a negative number, not-a-number, or –∞: 0 is returned. | | | | | | | | | | | |
| [Floating-point operation exceptions | Unimplemented operation exception (E) Invalid operation exception (V) Inexact exception (I) | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.



| CVTF.DU | W | Floating-point Convert Double to Unsigned-Word (Double) Conversion to unsigned fixed-point format (double precision) |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | CVTF.DUW reg2, reg3 | |
| [Operation] | $reg3 \leftarrow cvt reg2 (double \rightarrow w$ | word) |
| [Format] | Format F:I | |
| [Opcode] | 15 11 10 5 r r r r 0 1 1 1 1 1 1 reg2 | 4 0 31 27 26 25 23 22 21 20 17 16 1 0 1 0 0 w w w w 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 < |
| [Description] | This instruction arithmetically the register pair specified by a format, and stores the result in When the source operand is in result is outside the range of 2 detected. If invalid operation exception register is set as an invalid op follows, according to differen Source is a positive num Source is a negative num | y converts the double-precision floating-point format contents of general-purpose register reg2 to unsigned 32-bit fixed-point in general-purpose register reg3. infinite, not-a-number, or negative number, or when the rounded $2^{32} - 1$ to 0, an IEEE754-defined invalid operation exception is ns are not enabled, the preservation bit (bit 4) of the FPSR peration and no exception occurs. The return value differs as nces among sources. nber outside the range of $2^{64} - 1$ to 0, or $+\infty$: $2^{32} - 1$ is returned. mber, not-a-number, or $-\infty$: 0 is returned. |
| [Floating-point operation exceptions | Unimplemented operation exc] Invalid operation exception (V Inexact exception (I) | (V) |

[Operation result]

| reg2 (A) | +Normal | –Normal | I +0 −0 +∞ 0 (integer) Ma | | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|---------------------------------------------------------------------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.



| Floating-point Convert Double to Word (Double) | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------|--------|--------|--------|------|-----------------|----|------|---|-----|---|----|---|------|--------|------|-------|--------|------|---------------------------------|-------|------|-------|------|----|
| | | | | | | | | | | | | | | С | onve | ersior | n to | fixed | l-poin | t fo | rma | t (do | uble | oreci | sion | 1) |
| [Instruction format] | | CVI | rf.dv | V reg | g2, re | eg3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ← C' | vt reg | 2 (d | oubl | $e \rightarrow$ | wo | ord) | | | | | | | | | | | | | | | | | |
| [Format] | | Form | nat F: | Ι | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | 5 | Л | | | | 0 | 31 | | | 2 | 7 2 | S 25 | | 2.2 |))))))) | 1 20 | | 1. | 7 1 | 6 |
| | r r | r | r 0 | 1 1 | 1 | 1 3 | 1 1 | 0 | 0 | 1 | 0 | 0 | w | w | w | w v | , 1 | . 0 | 0 0 | 0 | 1 (|) 1 | 0 | 0 C | |) |
| | | reg2 reg3 category type sub-op | | | | | | | | | | | | | | | | | | | | | | | | |
| [Description] | reg2 reg3 category type sub-op This instruction arithmetically converts the double-precision floating-point format contents the register pair specified by general-purpose register reg2 to 32-bit fixed-point format, and stores the result in general-purpose register reg3. When the source operand is infinite or not-a-number, or when the rounded result is outside t range of 2 ³¹ – 1 to -2 ³¹ , an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are not enabled, the preservation bit (bit 4) of the FPSR register is set as an invalid operation and no exception occurs. The return value differs as follows, according to differences among sources. Source is a positive number or +∞: 2³¹ – 1 is returned. Source is a negative number, not-a-number, or -∞: -2³¹ is returned. | | | | | | | | | | the | f | | | | | | | | | | | | | | |
| [Floating-point operation exceptions | Unimplemented operation exception (E) Invalid operation exception (V) Inexact exception (I) | | | | | | | | | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CVTF.HS | | | | | | | | | | | | | | | | | | | Flo | oatii | ng-p | oin | t C | onve | ert H | lalf | to S | Single | ∍ (Si | ngle | ;) |
|-----------------------------------------|----|------------------------|---------------------------------|----------------------------|----------------------|--------------------------|----------------------------|-----------------------------|------------------------------|-------------------------|--------------------|-------------------|-----------------------|-----------------------|--------------------|--------------------|-------------------|---------------------|----------------------|----------------------|---------------------|--------------------|-------------|-----------------------|---------------|---------------------|--------------------|-----------------------|------------------------|----------------|--------|
| | | | | | | | | | | | | | | | | | Co | nve | rsio | n to | o flo | atin | g-p | oint | for | mat | (sir | igle p | oreci | sior | ı) |
| [Instruction format] | | C | VTF | F.HS | S re | g2, | reg | g3 | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | re | g3 ↔ | — c | vt re | eg2 | (ha | alf – | → si | ng | gle) |) | | | | | | | | | | | | | | | | | | | |
| [Format] | | Fo | orma | at F | :I | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | | 11 | 10 | | | | | - | 1 | | | | 0 | 24 | | | | 27 | 26 | 25 | | 22 | 22 | - 04 | 20 | | 1 | 7 1 | 6 |
| | r | r r | r | r | 1 | 1 | 1 | 1 | 1 1 | , L | 4 | 0 | 0 | 1 | 0 | w | w | w | W | <u></u> w | 1 | 0 | 0 | 23 | 1 | 0 | 0 | 0 | 0 : | | |
| | | rov | ~? | | 1 | | | | | - | - | - | - | | - | | | rog | 2 | | l | | too | 1011 | h / | | 1 | aub | <u></u> | | |
| [Description] | I | Th the rot ge | nis in e lov undi mera | nstr wer ing al-p | ructi 16 the | on bits res ose | arit s of ult reg | hme ger in a giste | etica iera cco r re | all 1-p rda g3 | y c ourj anc | onv pos e v | ver se r witl | rts t regi h th | he ste: ie c | hal r re uri | f-p g2 rent | to to | isic sinį und | on f gle- ling | loa pre g m | ting cis ode | g-p ior | oint n flo nd s | t fo oatii | rma ng- res 1 | at c poi the | onte nt fc resu | ents orma ılt ir | in at, 1 | 1 |
| [Floating-point operation exceptions |] | In | valio | d oj | pera | tio | n ex | cep | otion | ı (| V) | | | | | | | | | | | | | | | | | | | | |
| [Supplement] | | W be wi | ith t acc 11 n | the tura ot b | exce tely e fl | epti co ush | on (nve ed (| of n rted evei | ot-a l int 1 if | -n o s the | um sing e F | ibe gle S b | er va -pr bit o | alue ecis | es, sio he | all n fl FP | hal oat SR | lf-p ting reg | rec g-pc giste | isic oint er i | on f foi s 1. | loat ma | ting t v | g-po alue | oint es. 1 | t foi A s' | rma ubr | it va iorm | lues nal ir | ca npu | n t |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|----|----|----|----|-------|--------------|
| Operation result [exception] | A (ŀ | Half) | +0 | -0 | +∞ | _∞ | Q-NaN | Q-NaN [V] |

Note: [] indicates an exception that must occur.



| CVTF.LD | | | | | | | | | | | | | | Flc | oatir | ו-g | ooir | nt Co | onve | ert l | -on(| g to | Do | uble | : (Do | bub | le) |
|--------------------------------------|-----|------------------------------------------------|------------------------------------------|---------------------------------------|---------------------------------|---------------------------|-----------------------------|---------------------------|----------------------|----------------------|-------------------|-----------------|---------------------|---------------------|---------------------|--------------------|----------------------|----------------------|----------------------|------------|---------------------|--------------------|---------------------|----------|--------------------|--------------------|----------|
| | | | | | | | | | | | | C | Conv | /ers | ion | to | floa | ting | -poi | nt f | orm | at (| dou | ble | prec | isic | on) |
| [Instruction format] | | CVTF.L | D reg | 2, reg3 | 3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | cvt reg | g2 (lon | g-wo | ord | \rightarrow | doı | uble | e) | | | | | | | | | | | | | | | | | |
| [Format] | | Format | F:I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 1 | 1 10 | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 | 21 | 20 | | | 17 | 16 |
| | r r | rr (|) 1 1 | L 1 1 | . 1 | 1 | 0 | 0 | 0 | 0 | 1 | w | w | w | w | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | | reg2 | | | | | | | | | | | r | eg3 | | | | ca | tego | ory | typ | ре | | sub· | -op | | |
| [Description] | | This inst pair spec accordan by gener | tructio cified l nce wi ral-pur | n arith by gen th the pose r | meti eral- curre egist | cal pui ent er i | ly c rpos rou reg2 | conv se r ndi 3. | vert egi: ng 1 | ts th ster mod | ie (re le, | 64- g2 an | bit to c d st | fixe lou core | ed-j ble s tł | poi -pr ne i | nt f reci resu | forr sio ılt i | nat n flo n th | con oat | nteı ing regi | nts -po .ste | of t int r pa | the form | regi mat pec | iste in ifie | er ed |
| [Floating-point operation exceptions | ;] | Inexact | except | ion (I) | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | _ | _ | | _ | | | _ | _ | _ | _ | | _ | _ | _ | _ | _ | | | | | | |
| | | reg2 (A) Operation result [exception | on on] | +Inte | ger | | A | (No | –In rma | tege I) | ər | | | | 0 + | (in 0 | teg | er) | | | | | | | | | |



.

| CVTF.LS | | | | | | | | | | | | | | F | 102 | itinę | g-po | oint | Con | iver | t Lo | ng 1 | to S | ingle | e (Sin | gle) |
|--------------------------------------|-----|---------------------------------------------------|-------------------------------------|----------------------------------|----------------------------------|-----------------------|----------------------|---------------------|------------------------|-------------------------|------------|-------------------|---------------------|---------------------|--------------------|-------------------|--------------------|-------------------|-------------------|-------------------|----------------------|-------------------|---------------------|------------------------|-----------------------|------------------|
| | | | | | | | | | | | | (| Cor | ver | sio | n to | floa | ating | g-pc | oint | forn | nat | (sin | gle p | recis | ion) |
| [Instruction format] | | CVTF.L | S reg | , re | g3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | $reg3 \leftarrow c$ | evt re | g2 (l | ong-v | vord | $ \rightarrow$ | sin | gle) | | | | | | | | | | | | | | | | | |
| [Format] | | Format I | F:I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 1- | 1 10 | | | 5 | 1 | | | 0 | | 31 | | | | 27 | 26 | 25 | | 23 | 22 | 21 | 20 | | 17 | 16 |
| | r 1 | rr c | 1 | 1 1 | 1 1 | 1 | 0 | 0 | 0 | 0 1 | | w | w | w | w | w | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 1 | 0 |
| | | reg2 | | | | | | | | | | | r | eg3 | | | | ca | tego | ory | typ | pe | | sub- | р | |
| [Description] | | This inst pair spec stores the current r | ructio rified e resu oundi | on ar by g ilt in ing n | ithme enera gener node. | tical l-pu al-p | lly o rpo ourp | conv se r ose | verts regis regi | s the ter 1 ister | e 6 reş | 64-1 g2 eg3 | bit to s 3. T | fixe sing 'he | ed- gle- res | poi pre ult | nt ecis is 1 | forr ion ou | nat flo nde | co ati d ii | nter ng-j n ac | nts poi coi | of 1 nt f dai | the r form nce י | egis at, a with | ter nd the |
| [Floating-point operation exceptions | ;] | Inexact e | excep | tion | (I) | | | | | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | _ | | | | | | | | | | | _ | | | | | | | | | | | |
| | | reg2 (A) | | +In | nteger | | | | -Int | ege | r | | | | 0 | (in | teg | er) | | | | | | | | |
| | | result [exceptic | n] | | | | А | (110 | imal |) | | | | | + | Ū | | | | | | | | | | |



| CVTF.SD | | | | | | | | | | | | | | | FIO | atın | g-p | Sint | Cor | vei | rt S | ingle |) to | Doi | aple | e (D | duc | le) |
|------------------------------------------|----------|----------------------------------------|----------------------------|----------------------------------------|--------------------------------------|---------------------------|----------------------|----------------------|----------------------|--------------------|-----------------------|--------------------|---------------------|--------------------|----------------------|----------------------|-------------------|---------------------------|----------------------|--------------------|--------------------|-----------------------|-------------------|-------------------|------------------|-----------------------|-------------|-----|
| | | | | | | | | | | | | | (| Con | vers | sion | to | floa | ting | poi | nt f | orm | at (| doul | ble | pred | cisio | วท) |
| [Instruction format] | | CVTF | SE | reg2 | l, reg | 3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | - c | vt reg2 | 2 (sin | ıgle - | \rightarrow (| lou | ble) |) | | | | | | | | | | | | | | | | | | |
| [Format] | | Forma | t F | :I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 | 21 | 20 | | | 17 | 16 |
| | r | r r r | r | 1 1 | 1 : | 1 1 | 1 | 0 | 0 | 0 | 1 | 0 | w | w | w | w | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| | L | reg2 | | | | | | | | | | | | | reg | 3 | | | cat | ego | ory | typ | e | : | sub | -op | | |
| [Description] | | This ir genera the cur purpos | istr 1-p rren e r | uction urpose nt rour egister | n arith e regi nding r reg3 | imet ster moc 3. | ical reg le, : | lly o 2 to and | con o do l stc | ver oub ores | ts t le-p s the | he preo e re | sin cisi esul | gle on lt ir | -pro floa 1 th | ecis atin e re | ion g-p gis | flo flo foir ter | pati nt fo pai | ng- orm r sp | poi iat, pec | int f in a ifie | orr acc d b | nat ord y g | co anc ene | nter ce w eral- | nts vith | of |
| [Floating-point operation exceptions] |] | Unimp Invalio Inexac | oler l oj t ez | nentec peratic xceptio | d ope on exe on (I) | ratio cepti) | on e: ion | xce (V) | ptic) | on (| (E) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|---------|---------|----|----|----|----|-------|--------------|
| Operation result [exception] | A (Do | ouble) | +0 | -0 | +∞ | _8 | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| CVTF.SL | Floating-point Convert Single to Long (Single) |
|--------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Conversion to fixed-point format (single precision) |
| [Instruction format] | CVTF.SL reg2, reg3 |
| [Operation] | $reg3 \leftarrow cvt reg2 (single \rightarrow long-word)$ |
| [Format] | Format F:I |
| [Opcode] | |
| | 15 11 10 5 4 0 31 27 26 25 23 22 21 20 17 16 r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r r |
| | reg2 reg3 category type sub-op |
| [Description] | This instruction arithmetically converts the single-precision floating-point format contents of general-purpose register reg2 to 64-bit fixed-point format, in accordance with the current rounding mode, and stores the result in the register pair specified by general-purpose register reg3. When the source operand is infinite or not-a-number, or when the rounded result is outside the range of 2⁶³ - 1 to -2⁶³, an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are not enabled, the preservation bit (bit 4) of the FPSR register is set as an invalid operation and no exception occurs. The return value differs as follows, according to differences among sources. Source is a positive number or +∞: 2⁶³ - 1 is returned. Source is a negative number, not-a-number, or -∞: -2⁶³ is returned. |
| [Floating-point operation exceptions | Unimplemented operation exception (E) Invalid operation exception (V) Inexact exception (I) |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | 8 | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|---|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | -Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CVTF.SH | | | | | | | | | | | | | | F | loati | ng-p | point | Co | nve | rt Sing | gle to | o Hal | lf (Sir | ngle) |
|------------------------------------------|-----|-------------------------------------------------|----------------------------------|------------------------------------------------|-------------------------------------------------|-------------------------------|-----------------------------|--------------------------|-----------------------------------|-----------------------------|---------------------|--------------------|-----------------------|------------------------|---------------------------|---------------|----------------------|----------------------|-------------------|-----------------|--------------------|------------------------|--------------------------|-----------------|
| [Instruction format] | | CVTF. | SH | [reg2 | 2, reg3 | | | | C | onve | ersio | n to | o hali | t-pre | CISIOI | <u>n flo</u> | atin | <u>g-po</u> | oint | forma | : (sir | igle p | oreci | sion) |
| [Operation] | | reg3 ← | - Z6 | ero-ex | tend (| cvt 1 | reg | 2 (| singl | e → | hal | f)) | | | | | | | | | | | | |
| [Format] | | Format | t F: | I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 44 | 10 | | | F | 4 | | | 0 | 24 | | | 07 | | 05 | | 22 | 22.24 | . 20 | | 1. | 7 46 |
| | r r | rr | r | 1 1 | 1 1 | 1 | 1 | 0 | 0 | 0 1 | 1 | w | w | w | w w | 1 | 0 | 0 | 0 | 1 0 | 0 | 0 | 0 1 | 0 |
| | | reg2 | | | | | | | | | | | r | eg3 | | | ca | tego | ory | type | | sub- | ор | |
| [Description] | | This in genera accord stored | stri l-pi anc in g | uctior urpos ce wit genera | n arith e regis h the c al-purj | neti ter r urre pose | cal reg2 ent : reg | ly 2 to rou gis | conv o hali undin ter re | erts f-pre g m g3. | the ecisi ode | sin ion . Tl | gle- floa he ro | prec ating esult | cisior g-poi t is z | n flo nt f | oati Torm -ext | ng-j nat, tenc | poi rou led | nt for indin | mat g th ord | t cor e res leng | ntent sult i th ai | s in n nd |
| [Floating-point operation exceptions] |] | Unimp Invalid Inexac Overfle Underf | len l op t ex ow lov | nente peratio kcepti excej w excej | d oper on exc on (I) ption (eption | ation eptic O) (U) | n ez on (| xce (V) | eption) | n (E) |) | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|----|----|----|------------|-------|--------------|
| Operation result [exception] | A (ŀ | Half) | +0 | -0 | +∞ | _ ∞ | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| CVTF.SU | L | | | | | | | | | | | | | | | C | Cor | Flo | oat ers | ting | g-p n to | oint | t Co sigr | nve | fixe | Sinę ed-į | gle to point | Ur o Ur | nsig ma | ned t (si | l-Lo | ong le pr | (Sin | igle) ion) |
|-----------------------------------------|----|-------------------------------------------------------------|------------------------------------------------------------------|-----------------------------------------------------------------------|-----------------------------------------|-------------------------------------------------------------------------|----------------------------------------------------|-----------------------------------------------------|----------------------------------------|---------------------------------------------------|------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------|-------------------------------------------------|-----------------------------------|-----------------------------------------------------|-------------------------------|---------------------------------|---------------------------------------------|----------------------------|---------------------------------------------|----------------------------------------------------------------|-----------------------------|-----------------------------------|------------------------------------------------|--------------------------|---------------------------------------------------------|--------------------------------------------------|-----------------------------------------------------------|--------------|-------------------------|----------------------------|-------------------------------------------|--------------------------------|
| [Instruction format] | | C۱ | VT] | F.SU | UL | . re | eg2 | l, re | eg3 | ; | | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg | g3 · | ← c | evt | reg | g2 | (sir | ıgl | e – | → u | ins | igr | ned | lo | ng | ;-W | VO 1 | rd) |) | | | | | | | | | | | | | | |
| [Format] | | Fo | orm | nat F | F:I | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | | | | | | | | _ | | | | | | • | | | | | | 07 | | | _ | 0.0 | | | | | | 4- | |
| | 15 | | | 11 | | 0 | 1 | 1 | 1 | 1 | 1 | 4 | 0 | 1 | | <u>,</u> | 0 | 3 | 1 | | | | 27 | 26 | 25 |) (| 23 | 1 | . 21 | 20 |) | 0 1 | 1/ | 16 |
| | T | L 1 | | | - | L _ | L | 1 | 1 | 1 | T | | 0 | 1 | . (| , | 0 | , v | v | w | w | w | w | | 0 | | 5 0 | 1 | 0 | | | · · | | 0 |
| | | reg | <u>j</u> 2 | | | | | | | | | | | | | | | | | r | eg | 3 | | | Ca | ate | gory | ty | pe | | รเ | o-du | р | |
| | | gel cun reg Wl res det If i reg fol | ner rrei gist her sult tec inv gist llov | ter r n the t is c teted. valid ter i: Sou: Sou: | e s out l oj s s acc rce | pos ndii 3. our tsid per et a core e is e is | ng rce le t ations a din a p a r | ope ope he : on e an i ng te pos: | era ran exc nv o d itiv | ar f and nge cep alio liff ve f | nd is of do ere nur nu | 2 to stc inf 2 ⁶ 2 ⁶ ns per nc nb mb | ini are are es er oer, | ite, ite, - 1 e no ion am out | no to to an on sid | ed es t-a 0, ena d g s id e | a-n an abl no sou | t in num leo e r nt | n t mb EE d, 1 xc ces ces | the centre the ceps. | xe re 75 e p otic e o r - | a-p egis or n 4-d ores on c f 2 ⁶ | ega lefii erv occu | tiv ned atio 1 t s re | ir sj e n in in on l Th to (| um val bit ne : | , in cific nber lid c (bit retu pr +0 | acc d b , or pper 4) rn v ∞: 2 | ord y g wh cati of valu 2 ⁶⁴ | the -1 | th exa FI diff | e rc cep PSR fers | n in urpo ound tion as urn | e ose ded i is ed. |
| [Floating-point operation exceptions | ;] | Un Inv Ine | nim vali exa | nple id o act e | me pe exc | ente rati ept | ed o ion tior | ope ex n (I | erat cej | tio1 ptio | n ez on (| kce (V) | epti) | ion | (E |) | | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| CVTF.SU | W | r | | | | | | | | | | | | | | Cor | Floa | atin | ng- | poir to u | nsia | nve | rt Si fixe | ing d-r | le to | Uns | sigr mat | ied- | -W | ord | (Sin | igle) |
|--------------------------------------|-----------|-----------------------------------------|--------------------------------------------------------------------|---------------------------------------------------------------------------|--------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------|------------------------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|--------------------------------------------------------|------------------------------------|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------|-------------------------------------|---------------------------------|--------------------------------------------|-----------------------------------------------|--------------------------------------------------|----------------------------------------|-------------------------------------------------|---------------------------------------------|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|--------------------------------------------|-------------------------------------|---------------------------------|
| [Instruction format] | | C | ZVT | FF.S | SU | W | reį | g2, | re | g3 | | | | | | 001 | IVCI | | | | 1019 | | | | Joint | 1011 | nat | (011 | <u>igr</u> | | | |
| [Operation] | | r | eg3 | ← | · CV | vt re | eg2 | 2 (s | ing | gle · | → I | uns | igr | ned | WO | rd) | | | | | | | | | | | | | | | | |
| [Format] | | F | orn | nat | F: | I | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | | | 10 | | | | | - | | | | | 0 | | | | | 0 | 7 0/ | | | | | 04 | 00 | | | 4- | |
| | 15 r : | r | r | r | 11 r | 10 | 1 | 1 | 1 | 1 | 5 1 | 4 | 0 | 1 | 0 | 0 | 31 w | W | 7 | w | <u>2</u> v v | 1 | 0 | 0 | 0 0 | 1 | 0 | 0 | C |) 0 | <u> </u> | 0 |
| | | re | əg2 | | | | | | | | | | | | | | | | re | eg3 | | | са | teç | gory | ty | ре | | su | ıb-o | p | |
| [Description] | | T g V re d It re f | This ene ene Vhe esul etec f inv egis ollo | ins eral eral en ti lt is cteo val ster ows So | stru-pu -pu he s o d. id · is , a | uctio urpo sou utsio ope set cco cc is | on ose ose urco de rat as rdi s a | ari re re o tho ior an ng po | ithi gis gis per e ra n ez n in to sit | met iter iter rang ang xce val dif ive | ical reg reg d is e of ptic id c fere | $\frac{11}{2} \text{ to}$ 2 to $3.$ 100 f 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} 2^{3} | con o u ini ² _ are rati es | nver insig ite, 1 - 1 t e no ion amo | not- o 0. t er and ong | he d 3: a-r , an nab l no sou the | sin 2-b 100 1E led ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 11C ex 1 1 (x 1 1 (x 1 x 1 x 1 x 1 x 1 (x 1 x 1 | gle it f nbe EH , th ace es. | e-p fix er, E7 ne pt | orec or 54- pre ion | isio poin neg defi serv occ | n fl nt fe ativ nec vatie urs. | oati orm e nu l inv on t Th Th | ing at, um val oit ie 1 | g-po , and iber, id o (bit retur | int : l sto or per- 4) (m v | for ore wh atio | main s the second seco | t c ie i the exc FI liff | ont resu e rc cept PSR fers | ents ilt i ound tion as | s of n ded i is ed. |
| | | | • | So | ur | ce is | s a | ne | ega | tive | e nu | mb | er, | , not | t-a- | nur | nbe | er, | or | -α | : 0 | is re | etur | ne | d. | | | | | | | |
| [Floating-point operation exceptions |] | U Iı Iı | Jnin nva nex | npl lid act | len op ex | nent berat kcep | ted tio | op n e on (| pera exc (I) | atio epti | on e ion | xce (V) | epti) | ion | (E) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.



| CVTF.SW | / | | | | | | | | | | | | | | | Floa | ting | J-pc | oint C | Conve | rt S | ingl | e to | Wo | rd (S | Sing | le) |
|--------------------------------------|-----|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------|------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-------------------------------------------------------|------------------------------------------------------------------|---------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------|-------------------------------------------------------------------|------------------------------------------------------------|---------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|--------------------------------------------------|---------------------------------------------------------------------------|-------------------------------------|------------------------------------------|----------------------------------------------|----------------------------------------------|----------------------------------------------|--------------------------|-----|
| | - | | | | | | | | | | | | | C | Cor | nvers | sion | to | fixed | d-poin | t for | ma | t (si | ngle | pre | cisio | on) |
| [Instruction format] | | CVTF. | SW | √ reg | 2, re | eg3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | - c | vt reg | 2 (s | ingle | $e \rightarrow f$ | WO | rd) | | | | | | | | | | | | | | | | | | |
| [Format] | | Forma | t F: | I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | 10 | | | F | 4 | | | | 0 | 24 | | | | 07 | 00 | 05 | 0 | | | | | | 47 | 40 |
| | r r | rr | r | 1 1 | 1 | 1 | 1 1 | 0 | 0 | 1 | 0 | 0 | 31 w | w | w | w | <u>27</u> w | 1 | 0 | 0 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | reg2 | | | | | | | | | | | | I | reg | 13 | | | cat | tegory | ty | /pe | | sub | o-op | | |
| [Description] | | This in genera purpos When t range o If inva registe follows • So • So | Istr l-p e ro the of 2 lid r is s, a our | uction urpos egiste sourc 2 ³¹ – opera set a ccorc ce is ce is | n ari e reg ce op 1 to ttior s an ling a po a ne | thm giste g3. -2^{31} in exc inva- to d sitiv gativ | etica er reg nd is , an 1 septic alid c iffere ve nu ve nu | lly 2 to inf IEE ons ope: enc mb | cor o 3 init EE7 are rati es a er c oer, | 2-bi e or 54-c on a amo or + | ts that the transformation of transform | ne ked t-a- abl no sou 2 ³¹ | sing l-pc -num d in led, exc urce - 1 nbe | gle int mb iva the cep es. is r, c | -pr t fc ber, lid e p otic re or - | recisormation of the formation of the f | where at , a where a and b are a are a and b are a are | n fle and en t ior atic urs. | sto sto he r exc on b The s re | ng-pc res th counc cepti- it (bi it (bi e retu turne | ed : con i t 4) rn v d. | for esu resu s de of valu | rma lt in ult i etec the ue c | t co n ge s ou cted FP: liffe | onter enera utsio I. SR ers a | nts al- de t 1s | of |
| [Floating-point operation exceptions |] | Unimp Invalid Inexac | olen l op t ex | nente peratio ccepti | d op on e on (| oerat xcer (I) | ion e otion | xce (V | epti) | on (| (E) | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| CVTF.UL | D | | | | | | | | | FI | oati | ng-p | oint (| Conv | ert | Unsi | gneo | d-L | ong to | Dοι | uble (| Dout | ole) |
|--------------------------------------|-----|-------------------------------------------------|-------------------------------------|-------------------------------------------|----------------------------------|---------------------|---------------------------|-------------------------------------|----------------------------------|----------------------------|---------------------|---------------------|-------------------------|----------------------|----------------------|-----------------------|------------------------|---------------------|-----------------------------|---------------------|-------------------------|-----------------------|------------|
| | | | | | | | | | | | C | Conv | rersio | n to | floa | ting- | poin | t fc | ormat (| doul | ole pr | ecisi | on) |
| [Instruction format] | | CVTF.U | ILD | reg2, r | eg3 | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | cvt re | eg2 (un | signe | ed l | on | g-wo | rd — | → dc | oubl | e) | | | | | | | | | | | |
| [Format] | | Format l | F:I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 1 | 1 10 | | | 5 | 1 | | | 0 | 31 | | | 27 | 26 | 25 | 2 | 2 | 22 21 | 20 | | 17 | 16 |
| | r 1 | rr (| 1 1 | 1 1 | 1 1 | 1 | 1 | 0 | 0 C | 1 | w | W | w w | 0 | 1 | 0 | 0 0 | 0 | 1 0 | 1 | 0 0 | 1 | 0 |
| | | reg2 | | | | | | | | | | re | eg3 | | | cat | egor | у | type | 5 | sub-o | р | |
| [Description] | | This inst register format in specified | tructi pair s n acc 1 by ; | on arith specific ordanc general | hmeti ed by e wit -purp | ical gei h th | ly ner ne o e ro | conve ral-pu currei egiste | erts t rpos nt ro r reg | he t e re und g3. | unsi egis ing | igne ter 1 mo | ed 64 reg2 ide, a | -bit to d nd s | fixe oub store | ed-p le-p es tl | ooin oreci ne re | t fo isio esu | ormat on flo ılt in t | con atin he r | itents g-po egist | s of t int ær p | the air |
| [Floating-point operation exceptions | 5] | Inexact | excej | ption (I |) | | | | | | | | | | | | | | | | | | |
| [Operation result] | | reg2 (A) |) | +Inte | eaer | | | - | Inte | aer | | | - t | 0 (in | itea | er) | | | | | | | |

| reg2 (A) | +Integer | -Integer | 0 (integer) |
|------------------------------------|----------|----------|-------------|
| Operation result [exception] | A (No | ormal) | +0 |



| CVTF.UL | S | | | | | Floa | ting-po | int Co | onve | rt Ur | nsigne | d-Lon | j to S | Single | (Sing | gle) |
|------------------------------------------|--------------------------------------------------------|----------------------------------------------------------------------------|---------------------------------------------|---------------------------------------|-----------------------------------|-----------------------|-------------------------------|---------------------------|------------------------|---------------------|------------------------------|----------------------------|-----------------------|---------------------------|---------------|------|
| | | | | | | (| Conver | sion t | o flo | ating | g-point | forma | t (sir | igle pr | ecisio | on) |
| [Instruction format] | CVTF.UL | S reg2, reg3 | | | | | | | | | | | | | | |
| [Operation] | reg3 ← cv | vt reg2 (unsign | ed long | -word | \rightarrow si | ngle |) | | | | | | | | | |
| [Format] | Format F:J | I | | | | | | | | | | | | | | |
| [Opcode] | 15 11 | 10 | 54 | | 0 | 31 | | 27 | 7 26 | 25 | 23 | 22 2 | 1 20 | | 17 | 16 |
| | rrrr 0 | 1 1 1 1 1 | 1 1 | 0 0 | 0 1 | w | w w | w w | 1 | 0 | 0 0 | 1 0 | 0 | 0 0 | 1 | 0 |
| | reg2 | | | | | | reg3 | | | cat | tegory | type | | sub-o | p | |
| [Description] | This instru register pa format, an accordance | action arithmet air specified by d stores the re- e with the curr | ically c genera sult in g rent rou | onvert Il-purp general nding | s the ose re l-purp mode | unsi egist oose | gned 6 ter regi registe | 54-bi 2 to s er reg | t fix sing g3. 7 | ed-ј le-р Гhe | point : precisi result | forma on flo t is ro | t con atin unde | ntents g-poir ed in | s of ti nt | he |
| [Floating-point operation exceptions] | Inexact ex | ception (I) | | | | | | | | | | | | | | |
| [Operation result] | reg2 (A) | +Integer | | –In | teger | | | 0 (i | nteg | jer) | | | | | | |

| reg2 (A) | +Integer | –Integer | 0 (integer) |
|------------------------------------|----------|----------|-------------|
| Operation result [exception] | A (No | ormal) | +0 |



| CVTF.UW | /D |) | | | | | | | | Flo | oati | ng-l | poin | t C | onv | ert l | Unsi | gnec | -W | ord | to D | out | ole (C | Dout | ole) |
|--------------------------------------|-----|-----------------------------------------------------------------|--------------------------------------------------|---------------------------------------------------------|-------------------------------------|------------------------------|--------------------------------|----------------------|-----------------------|------------------|----------------------------|-----------------------------|------------------------------|-----------------------------|---------------------|---------------------------|---------------------|--------------------------------|---------------------|-----------------------------|-----------------------------|-------------------|------------------------|--------------------|------|
| | | | | | | | | | | | (| Con | vers | sion | to | floa | ting- | poin | i fo | rma | t (do | ubl | e pre | ecisi | on) |
| [Instruction format] | | CVTF.UV | WD re | eg2, reg | <u>3</u> | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← c | vt regź | 2 (unsig | gned v | wor | ·d — | → de | oubl | le) | | | | | | | | | | | | | | | |
| [Format] | | Format F | :I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 11 | 10 | | 5 | Λ | | | | 0 | 21 | | | | 27 | 26 | 25 | 2 | о ́ | 20.0 | 01 20 | ` | | 17 | 16 |
| | r : | rrrr | 1 1 | 1 1 | 1 1 | 1 | 0 | 0 | 0 | 0 | w | w | W | w | 0 | 1 | 0 | 0 (|) | 1 1 | 0 1 | , 0 |) 0 | 1 | 0 |
| | | reg2 | | | | | | | | | | | rega | 3 | | | cat | egor | y | type | ə | su | ıb-op |) | |
| [Description] | | This instr general-p the curren purpose r This conv | uction urpose nt rour egiste versior | a arithm e registe nding m r reg3. n operat | etica er reg node, tion is | lly o 2 to and s pe | conv o do l sto erfor | vert oubl ores | ts th le-pi the | ne rec rec | uns cisi esul ura | sign on lt in tely | ned floa n the 7, w | 32. atin e re ithe | -bit g-f egis | fix ooir ter any | ed- nt fo pai | poin orma r spe ss of | t fo t, i cif | orm n ac fied reci | at cor cor by sion | ont daı ger | ents nce v neral | s of with l- | 1 |
| [Floating-point operation exceptions |] | None | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | _ | | | | | _ | _ | | | | | |

| reg2 (A) | +Integer | -Integer | 0 (integer) |
|------------------------------------|----------|----------|-------------|
| Operation result [exception] | A (No | ormal) | +0 |



| CVTEUM | VS | | | | | | | | | | | Floa | ating | g-pc | oint | Cor | iver | t Un | sig | ned | l-Wo | ord | to S | ingl | e (S | ingl | e) |
|--------------------------------------|----|-----------------------------------------------|-------------------------|-------------------------------|------------------|----------------------|--------------------|---------------------|--------------------|-----------------------|-----------------|--------------------|----------------------|---------------------|-------------------|---------------------|--------------------|------------|---------------|---------------------|--------------------|-------------|---------------------|---------------------|---------------|---------------------|---------|
| | | | | | | | | | | | | | Cor | nve | rsio | n to | floa | ating | g-po | oint | forn | nat | (sin | gle | prec | isio | n) |
| [Instruction format] | | CVTF.UV | VS r | eg2, re | eg3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← cv | vt reg | g2 (uns | signe | ed v | VO | rd — | → si | ingl | e) | | | | | | | | | | | | | | | | |
| [Format] | | Format F: | I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | 11 | 10 | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 | 21 | 20 | | | 17 [·] | 16 |
| | r | rrrr | | | | T | | 0 | 0 | 0 | 0 | W | W | W | W | W | Ţ | 0 | 0 | 0 | | 0 | 0 | 0 | 0 | Ţ | 0 |
| | | reg2 | | | | | | | | | | | I | reg | 3 | | | cat | teg | ory | typ | pe | | sub | -ор | | |
| [Description] | | This instr general-p general-p mode. | uctio urpos urpos | n arith se regi se regi | ster i ster i | ical reg2 reg3 | ly 2 tu 3. ' | con o sin The | ver igle res | ts tl e-pr sult | he eci is | uns isio rou | sign on fl nde | ied loat ed i | 32 ting n a | -bit ;-pc cco | fix oint rda | ed- for | po ma w | int it, a ith | forn ind the | mat stor | t co res rrer | nte the nt ro | nts (resi | of ult i ling | in g |
| [Floating-point operation exceptions | 5] | Inexact ex | cept | ion (I) |) | | | | | | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | reg2 (A) | | +Inte | ger | | | | -11 | nteg | ger | | | | (|) (ir | teg | jer) | | | | | | | | | |
| | | Operation | | | | | F | ۹ (No | orma | al) | | | | | - | +0 | | | | | | | | | | | |

R01US0143EJ0100 Rev.1.00

Mar 05, 2015

result [exception]



| |) | | | | | | | | | | | Float | ing- | poin | t Cor | vert | Word t | o Doi | uble | (Dout | ole) |
|--------------------------------------|-----------|--------------------------------------------------------------------|------------------------------------------------------|--------------------------------------------|------------------------------|-------------------------|--------------------------------|---------------------------|---------------------|-----------------------------|-------------------------------|-------------------------|---------------------|----------------------------|-----------------------------------|--------------------------------|----------------------------------------|----------------------------------|-----------------------|-------------------|---------|
| ••••• | | | | | | | | | | C | Conv | /ersio | n to | floa | ting-p | point | format | (doul | ole p | recisi | on) |
| [Instruction format] | | CVTF.W | D reg2, | , reg3 | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← c | vt reg2 | (word - | → d | oubl | e) | | | | | | | | | | | | | | |
| [Format] | | Format F | I | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | _ | _ | | | | | | | | | | | | | | | |
| | 15 r 1 | <u>11</u> r r r r | 10 1 1 | 1 1 1 | 5 1 | 4 | 0 0 | 0 | 0 | 31 w | w | w w | 27 0 | 26 1 | 25 0 | 23 0 0 | 1 0 | 20 1 | 0 0 | 17) 1 | 16 0 |
| | | reg2 | | | | | | | | | r | eg3 | | | cate | gory | type | | sub-c | р | |
| [Description] | | This instr purpose re current ro register re This conv | uction a egister f unding eg3. gersion o | arithmet reg2 to mode, a operatio | tical dou and on is | lly co ble-j stor | onve preci es th form | erts t isior ie rea | he i flo sult | 32- pati t in urat | bit t ng-j the tely, | fixed point regis | -poi for ster | int f mat pair an | form t, in t r spe y los | at co acco cifie s of | ontents rdance ed by g precis | s of g e wit gener ion. | gene h the al-p | ral- ? urpo | se |
| [Floating-point operation exceptions | 5] | None | | | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | | | | | | | | |

| reg2 (A) | +Integer | –Integer | 0 (integer) |
|------------------------------------|----------|----------|-------------|
| Operation result [exception] | A (No | ormal) | +0 |



| CVTEWS | 5 | | | | | | | | | | | | | | | Floa | ating | g-pc | oint (| Cor | nver | t Wo | ord | to S | Sing | le (s | ingl | le) |
|------------------------------------------|---------|----------------------------------------|----------------------|---------------------------|---------------------------|------------------------|------------------------|---------------------|-------------------|--------------------|-----------------------|-----------------|--------------------|--------------------|---------------------|--------------------|-------------------|---------------------|---------------------|-------------------|----------------------|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|------------------------|-------------------|-----------|---------|
| | | | | | | | | | | | | | | Co | nve | rsio | n to | floa | ating | g-po | oint | form | nat (| (sin | gle | prec | isio | on) |
| [Instruction format] | | CVTF. | WS | S reg | 2, reg | g3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | - сง | vt reg | ;2 (w | ord | \rightarrow s | ing | le) | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format | : F: | I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 r | rrr | 11 r | 10 | 1 | 1 | <u>5</u> | 4 | 0 | 0 | 0 | 0 | 31 | 747 | 747 | 747 | 27 | 26 1 | 25 0 | 0 | 23 0 | 22 | <u>21</u> | <u>20</u> | 0 | 0 | 17 · | 16 0 |
| | | reg2 | Ŧ | | | | | | | 0 | | 0 | | | reg: | 3 | | - | cat | eg | ory | typ | be | ; | sub | -op | | 0 |
| [Description] | | This in purpose general mode. | stri e re l-pi | uction egiste urpos | n arit er reg e reg | hme ;2 to ;iste: | etica sing r reg | lly gle- 3. 7 | con pre The | ver cisi res | ts tl on : sult | he flo is | 32- atir rou | bit ng-j nde | fix poir ed i | ed- nt f n a | poi orn cco | nt f nat, rda | forn and ince | nat d s v w | t co tore rith | nter es th the | its of the rection of | of g esu rer | gen ilt in it ro | eral n ounc | - lin; | g |
| [Floating-point operation exceptions] |] | Inexact | t ex | cepti | ion (I | D | | | | | | | | | | | | | | | | | | | | | | |
| [Operation result] | | 10000 | • • | | . 1.4 | | | _ | | | | | _ | _ | _ | |) /: | 40.0 | | | _ | | | | | | | |
| | | reg2 (/ | -) | | +inte | eger | | | ()) | -11 | nteg | jer | | | | |) (ir | iteg | ler) | | | | | | | | | |

| reg2 (A) | +Integer | –Integer | 0 (integer) |
|------------------------------------|----------|----------|-------------|
| Operation result [exception] | A (No | ormal) | +0 |



| DIVF.D | | | | | | | | | | | | | | | | | | | | Floa | ating | -po | int D |)ivio | de ([| Jouł | ole) |
|-----------------------------------------|----|----------------------------------------------|------------------------------------------------------|------------------------------------------------------|------------------------------------------------------|------------------------------------------------|------------------------------------|-------------------------------------|-----------------------------------------|----------------------------------------|--------------------------------------|-----------------------------------|-----------------------------|------------------------------------|------------------------------------|----------------------------------|-------------------|--------------------------------|-------------------------------------|---------------------------------------|------------------------------------|------------------------------------|--------------------------------------|------------------------------|----------------------------------------|-------------------------------------|----------|
| | | | | | | | | | | | | | | | | | FI | oati | ng-l | point | divis | sion | (do | ubl | e pre | ecisi | on) |
| [Instruction format] | | DIV | F.D | reg1, | reg2, | , reg3 | 3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ← re | eg2 ÷ | reg1 | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Form | nat F | :I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | 2 | 3 2 | 22 | 1 20 |) | | 17 | 16 |
| | r | r r | r 0 | 1 1 | 1 | 1 1 | 1 | R | R | R | R | 0 | w | W | W | w | 0 | 1 | 0 | 0 | 0 1 | . 1 | 1 | 1 | L 1 | 1 | 0 |
| | | reg2 | | | | | | | I | reg1 | | | | I | reg3 | | | | ca | tegor | y t | ype | | รเ | np-ob |) | |
| [Description] | | This spec cont the r were mod | instr ified ents o regist e of in e. | uction by ge of the er pai nfinite | n divi neral regis r spece accu | ides o l-purj ster p cifieo uracy | dou pos air d by 7, ar | ible e re spe y ge nd t | e-pre egist cific ener he r | ecisi ter r ed b al-p resu | ion reg2 by g burp lt is | flc 2 bj gen bos 5 rc | oati y tl era e ro | ng- he c ll-p egis dec | poi lou urpo ster l in | nt i ble ose reg aco | for -pr g3. | ma eci gist Th dar | t co sion ter 1 e o nce | nten floa reg1, pera with | ts o atin, and ion the | f th g-p l sto is o cu | e re oint ores exec rren | gis fo th cut tr | ster orma ne re ted a roun | pain at sult is if ding | in it |
| [Floating-point operation exceptions |] | Unir Inva Inex Divi Ove: Und | npler lid oj act e: sion rflow erflov | nente perati xcepti by zer exce w exc | d ope on ex ion (I ro exe ption eptio | eratic cepti () cepti (O) on (U | on e ion on (| (V) | eptic) | on (| E) | | | | | | | | | | | | | | | | |



[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|-------|--------|-------|------------|-------|-----------|
| reg1(A) | Normal | -Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| +Normal | | В - | ÷ A | | +∞ | _∞ | | |
| -Normal | | | | | -∞ | +∞ | | |
| +0 | | [7] | | | +∞ | -∞ | | |
| -0 | ±~~ | [4] | Q-INd | מא [א] | _∞ | +∞ | | |
| +∞ | +0 | -0 | +0 | -0 | | | | |
| _∞ | -0 | +0 | -0 | +0 | Q-112 | מיי ניין | | |
| Q-NaN | | | | | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| DIVF.S | | | | | | | | | | | | | | | | | | | | FI | loat | ting | -poi | int E | Divid | 3) et | Sing | gle) |
|-----------------------------------------|-----|--------------------------------------------------------------|-----------------------------------------|----------------------------------------------------------|---------------------------------------------------|-------------------------------------------------|-----------------------------------|--------------------------|------------------------------|-------------------------------|--------------------------------|--------------------------|-----------------------------|-----------------------------|--------------------|--------------------------|--------------------|---------------------------|-----------------------|--------------------------|-----------------------------|------------------------------|------------------------------|----------------------------|----------------------------|------------------------------|------------------------------|------------------|
| | | | | | | | | | | | | | | | | | F | loat | ing- | poii | nt d | livis | sion | (sir | ıgle | pre | cisi | on) |
| [Instruction format] | | DIVF.S | 5 re | eg1, | reg2 | , reg | 3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | - re | g2 ÷ | regl | l | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format | F: | I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | 5 | | | | | 0 | 21 | | | | 07 | 26 | 25 | | 22 | 22 | 0.01 | 20 | | | 17 | 16 |
| | r i | rr | r | 1 1 | . 1 | 1 | 1 1 | - R | R | R | R | R | w | w | w | W | w | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| | | reg2 | | | | | | Ì | | reg | 1 | | | r | eg3 | | | | cat | iego | ory | ty | pe | | sub | o-op | | |
| [Description] | | This ins register reg1, an were of mode. | stru re nd f in | iction g2 by store finite | n div y the s the e acc | rides sing rest curac | the gle-p ult i | sin orec n g nd | gle- cisio ener the | pre on fl ral-j resu | ecisi loat purj ult i | on ing pos s re | flo g-pc se re oun | atin oint egis ded | g-p forr ter | oin nat reg acc | t fo cc 3. ' | orn onte The dar | nat o ents e op | con of pera wit | iten gen atio h tl | nts o ner on i he o | of g al-p is e: cur | gene ourj xec ren | eral 205 ute t ro | l-pu e re d as ounc | urpo egis s if ling | ose ter it |
| [Floating-point operation exceptions |] | Unimpl Invalid Inexact Divisio Overflo Underf | lem op ex ex on b ow lov | nente peration ception py zen exce v exce | d op on e: ion (ro ex ption eptio | erati xcep (I) xcept n (O) on (U | ion o otion tion) J) | exc 1 (V (Z) | epti ')) | on | (E) | | | | | | | | | | | | | | | | | |



[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|------|-------|-------|---------|-------|-----------|
| reg1(A) | Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
| +Normal | | В - | ÷Α | | +∞ | _∞ | | |
| -Normal | | | | | _∞ | +∞ | | |
| +0 | ±∞ | [Z] | Q-Na | N [V] | +∞ | -∞ | | |
| -0 | | | | | _∞ | +∞ | | |
| +∞ | +0 | -0 | +0 | -0 | | | | |
| -∞ | -0 | +0 | -0 | +0 | Q-112 | נוא [V] | | |
| Q-NaN | | | | | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.


| FLOORF. | D | | | | | | | F | loati | ng- | poin | nt C | on | ver | t D | oub | le to | Lo | ng, ı | ound | towa | ard i | neg | ative (| (Dou | ble) |
|--------------------------------------|-----------|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------------------------------------|---------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------------------------------|----------------------------------------------------------|---------------------------------------------------------------|-----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|-------------------------------------------------------------|-------------------------------------------------------------|-----------------------------------------------------------------------------|--------------------------------------------------------------|----------------------------------------------------------|---------------------------------------------------|-----------------------------------------------------|------------------------|-------------------|
| | | | | | | | | | | | | | | С | or | vers | sion | to fi | xed | -point | form | nat (| (dou | iple bi | recis | ion) |
| [Instruction format] | | FLO | ORF | .DL r | eg2, 1 | reg3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ← fl | oor re | g2 (d | oubl | e – | → lo | ong- | wo | ord) | | | | | | | | | | | | | | | |
| [Format] | | Form | at F | :I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | 10 | | | _ | | | | | • | ~ 4 | | | | | ~~~ | | | | | | | | |
| | 15 r 1 | rı | <u>11</u> r 0 | 10 | 1 1 | L 1 | 5 1 | 4 | 0 | 0 | 1 | 1 | 31 w | w | V | <i>i</i> w | 0 | 1 | 25 | 0 0 | 1 | <u>21</u> 0 | 1 | 0 1 | 1/ L 0 | 16 |
| | | reg2 | | | | | | | | | | | | | re | g3 | | | ca | tegory | ty | ре | | sub-c | р | |
| [Description] | | This the restores stores The r When range If inv regiss follow | instr egiste s the result n the e of 2 valid ter is ws, a Sour Sour | uction result t is rou sourc $2^{63} - 1$ opera s set as accord rce is a rce is a | arith spec in th undec e ope to -2 tion e s an in ing to a posi | meti ified e reg l in the rand 2 ⁶³ , a exception vali o diff tive to the | call by gisto he is i an I otio d o ere nur nur | ly c ge er p -∞ (infi EE ns a per nce mbe | conv nera pair : dired nite EF75 are 1 ex ar ex ar er or er, n | ert il-p spe ctic or 54-c not not not | s th purp ecifi on, 1 not defi ena nd 1 ng s ∞: 2 ∞: 2 | e co led reg -a- abl no sou 63 un | lou e r by gar nu d i ed ex urco | ible egi dle mb nva , th cep es. 1 is | e-p ste ss per ali- e j oti | orection er recent and the second se | ision eg2 -pun he of who pera erva poccu ned -2^{6} | n fl to (rpo curr en t tion atic urs. | oati 54-t se r rent he 1 n ex on b The | ng-po bit fix egiste round cound cepti- it (bit e retu | bint ed-1 er re ding ed r on i (4) rn v | for poin g3. g m esu s de of t valu | ma nt f odd ilt i eteo ihe ue d | t cont orma s outs cted. FPSF iffers | tent: t, ar side | s of id the |
| [Floating-point operation exceptions |] | Unim Inval Inexa | npler id op act ex | nenteo peratic xcepti | l oper on exc on (I) | ration | n ez on (| xce _j (V) | ptio | n (. | E) | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | 8 | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|---|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | -Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| | | | | | | | | | | | | | | | Cor | nve | ersi | ion | to | uns | sign | ed | fixe | d-p | point | fori | nat | : (dc | bub | le p | recis | sion |
|--------------------------------------|----|----------------------------------------------|-----------------------------------------------|-----------------------------------------------------|----------------------------------------|-----------------------------------------|---------------------------------|-------------------------------------------|------------------------------------|---------------------------------------|----------------------------------------|------------|-------------------------------|-------------------------------------------|---------------------------------------|-------------------------------|-----------------------|------------|---------------------------------|-----------------------------|---------------------------|-------------------|---------------------------|-------------------------|-----------------------------------------|-------|-----------------|----------------------------------|------------------------------|-----------------------------|---------------|------|
| [Instruction format] | | FL | .00 | ORF | .DU | L | reg | 2, 1 | reg. | 3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg | g3 ← | — fl | loor | reg | g2 (| dou | uble | e – | י ט | ınsi | gn | ed | lon | g- | wc | ord |) | | | | | | | | | | | | | |
| [Format] | | Fo | rma | at F: | :I | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | | | 11 | 10 | 1 | 1 | 1 | 1 | 1 | 4 | 0 | 0 | 1 | 0 | 3 | 31 | | | | 27 | 26 | 5 25 | 5 | 2: | 3 22 | <u>2 2</u> | 1 20 |) | 0 1 | 17 | 16 |
| | r | r r | r | 0 | 1 | Ţ | T | T | Ţ | T | T | 0 | 0 | Ţ | Ţ | <u> </u> | W | W | W | W | 0 | | 0 | | 0 0 | 1 | 0 | | | 0 1 | | 0 |
| | | reg | J2 | | | | | | | | | | | | | | | I | reg | 3 | | | C | ate | gory | ' ty | /pe | | SI | ub-c | p | |
| | | Th WI res det If i reg fol | e re hen sult tecte inva giste | the the is o ed. alid er is vs, a | t is 1 sou outsi ope s set | rou irce de rati as rdii | nde e op the ion an | ed i pera rar exc inv to c | n th and age cep ralid | ne - is of tio d o ere | $-\infty$ inf 2^6 ns per | are are | rec te, 1 t no on | tion not to (ot e and ong | n, re -a-1), a1 nab d no | eg nu n I ole o e | arc ml EI d, | the sep | ss r, c 75 e p otic | of n or n 4-d ores | the ega efin erv | cun tiv nec | rrer e n in on 1 | nt i un va bit | rour nbei lid d : (bit retu | t 4) | w rati | noc hen ion the ue (| le. th ex FI dif | ne ro cep PSI fers | ound otion | ded |
| | | • | S | Sour | ce i | s a | pos | sitiv | ve 1 | nur | nb | er o | out | side | e th | e 1 | rar | ıge | e o | f 2 ⁶ | 54 _ | 1 | io (|), (| or + | xo: 2 | 2 ⁶⁴ | - 1 | l is | s re | turn | ed. |
| | | • | S | Sour | ce i | s a | neg | gati | ve | nu | mt | ber, | no | ot-a | nu | ml | beı | r, c | or - | -∞: | 0 i | s re | etui | rne | ed. | | | | | | | |
| [Floating-point operation exceptions |] | Un Inv Ine | imp valio exac | pler d op ct ex | nent pera xcep | ted tion otio | ope n ex on (] | erat kcej I) | tior ptic | n ez on (| kce (V) | epti) | on | (E) |) | | | | | | | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| | | | Floating | g-poin | t Conv | ert Do | uble | to Unsi | gned | -Wo | rd, r | ound t | oward | d neg | jative | (Dou | ble) |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------|---------------------------------------------------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------|-----------------------------------|------------------------------------------------------------------------------------|----------------------------------------------------|---------------------------------------------------------------|--------------------------------------------------|----------------------------|-----------|
| FLOOKF | | | | | | Con | versi | on to ur | nsian | ed fi | xed- | noint f | orma | t (do | uble r | orecie | ion) |
| | | | | | | 0011 | VCISI | | ISIGIN | | Acu | point | onna | 1 (00 | | 0000 | |
| [Instruction format] | FLOORF | .DUW re | g2, reg3 | | | | | | | | | | | | | | |
| [Operation] | reg3 ← fl | oor reg2 | (double - | → un | signec | l wor | d) | | | | | | | | | | |
| [Format] | Format F: | Ι | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | |
| | 15 11 | 10 | 5 | 4 | | 0 | 31 | | 27 | 26 | 25 | 23 | 22 2 | 1 20 | | 17 | 16 |
| | rrrr O | 1 1 1 | 1 1 1 | 1 | 0 0 | 1 1 | W | www | w w | 1 | 0 | 0 0 | 1 (|) 1 | 0 | 0 0 | 0 |
| | reg2 | | | | | | | reg3 | | | cate | egory | type | • | sub-o | ор | |
| | the register format, ar The result When the result is o detected. If invalid register is follows, a • Sour | er pair spe nd stores t t is round source op utside the operation set as an according ce is a po | ecified by the result ed in the perand is range o e cange o n exception invalid to differ sitive nu | y gen t in g $-\infty$ d s infir f 2 ³² ons a opera ences mber | eral-p eneral- irection nite, no – 1 to re not tion an s amor | urpos -purp on, reg ot-a-n 0, an enabl nd no ng sou de the | e reg ose 1 gardl umb IEE ed, t exce urces | gister r register less of per, or r E754- the pre eption ge of 2 | r reg2 r reg the c nega defir serva occu | to u 3. curr tive ned atio urs. 1 to | ent ent inva n bi The | gned $\frac{1}{2}$ round mber, alid o at (bit retur or $+\alpha$ | 32-bi ling 1 or w perat 4) of m val | t fix mode then ion of the lue of $2^2 - 1$ | ed-po e. the r excep FPSI liffers | ounc otion R s as | led is |
| | • Sour | ce is a ne | gative nu | ımbe | r, not- | a-nun | nber | , or –∞ | : 0 i | s re | turn | ed. | | | | | |
| [Floating-point operation exceptions | Unimplen] Invalid op Inexact ex | nented op peration e exception (| eration e xception I) | excep (V) | tion (I | E) | | | | | | | | | | | |
| [Operation result] | | | | | | | | | | | | | | | | | |

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| FLOORF | D | W | I | | | | | | | F | loat | ing- | -poii | nt C | Con | ver | t D | oub | le to | o We | ord, | rou | und 1 | toward | neç | jative | : (Do | uble) |
|--------------------------------------|----|-----------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------|-------------------------------------------------------------------------|--------------------------|-----------------------------------------------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------------------------|------------------------------------------------------------------------------|-----------------------------------------------------------------------------|------------------------------------------------------------------------|---------------------------------------------------------|----------------------------------------------------------------------------|-------------------------------------------------|--------------------------------------------------------------|-------------------------------------------|---------------------------------------------------|-------------------------------------------------------|-------------------------------------------------------------------|----------------------------------------------------|----------------------------------------------------------------|--------------------------------------|----------------------|
| | | | | | | | | | | | | | | | | (| Cor | nver | sion | to | fixed | d-po | oint | format | (do | uble | prec | ision) |
| [Instruction format] | | FL | .00 | RF | .DW | re | eg2, | reg. | 3 | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg | <u>3</u> 3 ← | - f] | loor 1 | eg | 2 (d | loub | le – | → v | vorc | ł) | | | | | | | | | | | | | | | | |
| [Format] | | Fo | rmat | t F | :I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | | 11 | 10 | | | | 5 | А | | | | 0 | 31 | | | | 27 | 7 26 | : 25 | | 23 | 22.24 | 1 20 | h | 1 | 7 16 |
| | r | r r | r | r | 1 | 1 | 1 : | 1 1 | 1 | 0 | 0 | 0 | 1 | 1 | w | w | v | <i>z</i> w | | 1 | 0 | , 0 | 23 | 1 0 | 1 | 0 | 0 (| |
| | | reç | <u>j</u> 2 | | | | | | | | - | - | | | | | re | g3 | | | Са | ateg | gory | type | | sub- | ор | |
| [Description] | | Th the sto Th WI rar If i reg fol | is in reg ores t le res hen t nge c inval gister llow: So | str ist the sul the of 2 lid r is s, a our | uctio er pa resu t is ro sour 2^{31} – oper s set a accor rce is | n a ir s lt i our ce 1 t ations a din a p a n | arith spec n ge ondec on e an in ng to posi | imet ifiec eners l in t eranc 2^{31} , excep nval tive ative | ical l by al-p the l is an tio id o fere nun | ly o y ge ourp | conv energiose dire dire EE7: are ratio es a: er o: | vert al-j e rej e or 54- not mo r + not | ts th purj gist ion, def t en and ong ∞ : 2 | ne o pos er re t-a ine abl no sou 2 ³¹ | dou se r reg gau -nu ed i led o ex urco - nbe | uble regi g3. rdle umb nv; th acej es. 1 is | e-p iste ess ber ali ne j pti s re | orec er re of , or d op pres on c etur $-\infty$ | isio eg2 the wh pera serv occ | on fl to $\frac{1}{2}$ cur en atio vatio urs. | loat 32-1 the n ez on t Th | ing bit nt rou rou xce bit ne r | g-pc fixe oun unde eptic (bit retur | oint fo ed-poi ding r ed respon is d 4) of rn valu | rma nt f nod ult i lete the ue c | it con formation le. is out cted. FPS liffer | nten at, a tside R rs as | ts of nd e the |
| [Floating-point operation exceptions | 5] | Ur. Inv | nimp valid | leı oj | nente | ed (| ope | ratio cepti | n e | xce (V) | eptic | on (| (E) | | | , | | | | | | | | | | | | |
| | | Ine | exact | t e | xcept | ior | n (I) |) | | | | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| FLOORF. | S | L | | | | | | | | | Floa | ating | g-po | int | Co | onvo | ert | Dou | ble | to L | .ong | , rour | nd t | tow | ard | ne | gative | ∍ (Sir | ngle) |
|--------------------------------------|----|-------------------------------------|------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|---------------------------------------------------------------------------|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|-------------------------|----------------------------------------------------------------------------|---------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|----------------------------------------------------------------|----------------------------------------------------------------|------------------------------------------------------------|--------------------------------------------------------------------------|---------------------------------------------------------------|----------------------------------------------------|--------------------------------------------------------|--------------------------------------------------------------|------------------------------------------------------------|--------------------------------------------------|--------------------------------------------------|-----------------------------------------------|------------------------------|-------------------------------------|---------------------|
| | | | | | | | | | | | | | | | | | С | onve | rsio | n to |) fixe | d-poi | int | for | mat | (sir | igle p | precis | sion) |
| [Instruction format] | | F | LO | ORF | S.SL | reg | 2, re | eg3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | re | 2g3 | ← f | loor 1 | eg2 | 2 (si | ngle | \rightarrow | • lo | ng- | wo | rd) | | | | | | | | | | | | | | | | |
| [Format] | | F | orm | nat F | :I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | | 11 | 10 | | | | 5 | А | | | | 0 | 31 | | | | 2 | 7 2 | 5 25 | | 23 | 22 | 21 | 20 | | 17 | 7 16 |
| | r | r | r i | r r | 1 | 1 1 | 1 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | w | w | , | w v | 7 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 0 | 0 |
| | | re | ∋g2 | | | | | | | | | | | | | | re | g3 | | | са | itegoi | ry | ty | pe | | sub- | ор | |
| [Description] | | T gı T W ræ If fc | his a ener air s he r Vher inge inv egist ollo | instr ral-p spec: resul n the e of 2 valid ter is wws, a Sour | ructic iurpo ified t is ra sour coper s set a accor rce is | on a se r by ; oun ce (atic as a din, a p a n | rith egis gend ded oper on e an in g to posit | meti ster 1 eral- in t rand 2 ⁶³ , a xcep vali vali tive | cal reg2 pur he is i an l otio d o čere nur nu | ly o 2 to rpo | con o 64 dir inite EE7 are ratio es a er o oer, | ver 1-bi regi rect e or 554- no on a umo pr + not | ts that first fir | he re re t-a- ine abl no sou 2 ⁶³ nur | sin l-p ga ga -nu ed i led urc - nb | ngle oir rdl inv l, tl cce es. 1 i er, | e-r nt f ess be vali he pt: s r or | Form form s of r, or id o pre ion etur $-\infty$ | isio hat, the wh peraserv occ mec : -2 | n f and cu en atic ati urs I. | loati d sto rren the on ez on b . Th | ing-p pres t t rou roun kcept bit (b e ret | ooin the und de tion it ² urr | nt : re ling d r n is 4) o n v | forn sul g m esu s de of t alu | mat t in node lt is steec he l | e. s out rted. FPSJ | tent: regis side R s as | s of ster the |
| [Floating-point operation exceptions |] | U Ir Ir | nim val | npler id og act e | nente perat xcept | ed c ion tion | oper exc (I) | atio epti | n ex on | xce (V) | eptio) | on (| (E) | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | –Normal | +0 | -0 | +∞ | 8 | Q-NaN | S-NaN |
|------------------------------------|--------|---------|--------|-------|-----------------|---|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| FLOORF | Floating-point Convert Single to Unsigned-Long, round toward negative (Single) SUL Conversion to unsigned fixed-point format (single precision) |
|--------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [Instruction format] | FLOORF.SUL reg2, reg3 |
| [Operation] | reg3 \leftarrow floor reg2 (single \rightarrow unsigned long-word) |
| [Format] | Format F:I |
| [Opcode] | |
| | <u>15 11 10 5 4 0 31 27 26 25 23 22 21 20 17 16</u> |
| | r r r r r 1 1 1 1 1 1 1 1 w w w w 0 1 0 0 0 1 0 0 1 0 0 0 1 0 0 |
| | reg2 reg3 category type sub-op |
| [Description] | This instruction arithmetically converts the single-precision floating-point format contents of general-purpose register reg2 to unsigned 64-bit fixed-point format, and stores the result in the register pair specified by general-purpose register reg3. The result is rounded in the -∞ direction, regardless of the current rounding mode. When the source operand is infinite, not-a-number, or negative number, or when the rounded result is outside the range of 2⁶⁴ - 1 to 0, an IEEE754-defined invalid operation exception is detected. If invalid operation exceptions are not enabled, the preservation bit (bit 4) of the FPSR register is set as an invalid operation and no exception occurs. The return value differs as follows, according to differences among sources. Source is a positive number outside the range of 2⁶⁴ - 1 to 0, or +∞: 2⁶⁴ - 1 is returned. |
| [Floating-point operation exceptions | Unimplemented operation exception (E)] Invalid operation exception (V) Inexact exception (I) |

[Operation result]

| reg2 (A) | Normal | –Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.



| FLOORF. | S | U | M | | | | | | Flo | atir | ng- | poin | it C | on | ve | rt S | Sin | gle | to | Uns | sign | ed | I-W | ord | , ro | unc | l tov | va | rd ne | ega | ative | e (Si | ng | e) |
|-----------------------------------------|---------|------------------------------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------|------------------------------------------------------------------|-----------------------------------------------------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------|-----------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------|-------------------------------------|---------------------------------------------------------|------------------------------------------------------|--------------------------------------------------------|---------------------------------------------------------------|--------------------------------------------------------------------|-------------------------------------------------------|----------------------------------|--------------------------------------------------------------------|-------------------------------------------------------|--------------------------------------------------------|--------------------------------------------------------|------------------------------------------------|-------------------------------------------------------|--------------------------------------------------------------------------|------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------|--------------------------------------------|-----------------------------------------|--------------------------|--------------|
| | | | | | | | | | | | | | | | (| Cor | างต | ersio | on | to ι | insi | gn | ed | fixe | ed-p | oin | t foi | m | at (s | ing | gle p | reci | isio | n) |
| [Instruction format] | | F | LO | ORF | .SU | W | reş | g2, | reg | ;3 | | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | re | eg3 | ← f | loor | re | g2 (| (sin | gle | ; —) | • u | nsig | gne | ed | wo | ord | l) | | | | | | | | | | | | | | | | | |
| [Format] | | F | orn | nat F | :I | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | | 10 | | | | | _ | | | | | | | | | | | | - | ~~~ | ~- | | | | | | ~ | | | - | |
| | 15 r | r | r | 11 r r | 10 | 1 | 1 | 1 | 1 | 1 | 4 | 0 | 0 |) | 1 | 1 | 3 | 1 7 W | , . | w | w | 27 w | 1 | 25 | 0 | 23 | 1 | 2 | 0 0 | <u> </u> | 0 | 1) (| / · | 0 |
| | _ | - | - | | - | - | - | - | - | - | | Ū | | | _ | - | | | re | | | | - | | too | | - | | | 0 | ub | | | <u> </u> |
| | | | ,9z | | l | | | | | | l | | | | | | | | ic | ,go | | | | 00 | neg | jory | 5 | p | | 3 | ub | γ | ļ | I |
| [Description] | | T g T W ree d u If ree fc | his ene he v/he esul etec inv egis ollo | instr eral-p eral-p resul n the lt is c cted. valid ster is Sour Sour | ructi urpo urpo t is s sou outsi ope s set acco rce i | ion ose rou urco ide erat t as ordi is a | arit e reg e reg unde e op the tion an ing . pop | thm gist gist ed i pera ran ex inv to c siti | neti er r er r n tl and nge cep valid diff ve r | cal reg he is of otio d o ere nur | $\frac{1}{2} t$ $\frac{2}{3} t$ $\frac{-\infty}{2} t$ $\frac{2}{2} t$ $\frac{2}{3} t$ 2 | $conto uno dinfinit3^2 -arerati-ces aper co-ber,$ | rec te, 1 on amo out | erts gn etic no to ot ar on sic | s the on, ot-a 0, ena nd g s le | re 32 re a-n an abl no sou the | sin 2-1 ga iun 10 lec er urc er nb | ngle bit f mbe EEI d, th xce ces. ang oer, | e-p fix es er, E7 ne pt ge or | s of or 54- pre of 2 | cision point f the neg odef eser occ 2^{64} | on int e c gat Tin va cu - is | fle t fo cur tive ed atio rs. 1 t | oat: orm ren e nu inv on t Th o 0 | ing at, it roum vali oit (ne r | -pc and ber id c (bit etu r + 1. | oint d st din , or opper (24) rn (24) rn (22) | fc or g wrat of va 2^{3^2} | formation $rest formation for the formation for$ | at o he de. th ex F dif | con res ne r ccep PSI ffers | tent ult oun otio R s as | ts o in nde n i | of d s |
| [Floating-point operation exceptions |] | U Ir Ir | nin nva nexa | nplei lid oj act e | men pera xcej | ted tio ptic | l op n e: on (| era xce I) | tioı ptio | n ez on | xc (V | epti) | on | (E | E) | | | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (integer) | 0 [V] | 0 (int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| FLOORF. | SN | | | | | | Floa | ting | -poin | t Co | onve | ert | Sing | le to | W | ord, I | round | toward | l ne | gative | ∍ (Sir | igle) |
|---------------------------------------|------------------------------------|--------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|--------------------------------------------------------------------------|----------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|-------------------------------------------------|-----------------------------------------|----------------------------------------------------------------|-------------------------------------------------------------------|-------------------------------------------------------|--------------------------------------------|-------------------------------------|-------|
| | | | | | | | | | | | (| Co | nver | sion | to | fixed | -point | format | t (sir | ngle p | precis | ion) |
| [Instruction format] | F | LOORI | F.SW | reg2, reg | g3 | | | | | | | | | | | | | | | | | |
| [Operation] | re | eg3 ← f | loor re | eg2 (sing | gle — | • w | ord) | | | | | | | | | | | | | | | |
| [Format] | F | ormat F | :I | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 11 | 10 | | 5 | 4 | | | 0 | 31 | | | | 27 | 26 | 25 | 23 | 22 21 | 20 | | 17 | 16 |
| | r r | rrr | 1 1 | 1 1 | 1 1 | 0 | 0 | 0 | 1 1 | w | w | w | w | w | 1 | 0 | 0 0 | 1 0 | 0 | 0 | 0 0 | 0 |
| | re | eg2 | | | | | | | | | | reg | j 3 | | | cate | egory | type | | sub- | ор | |
| [Description] | T g T V ra If fo | his instruction eneral-purpose of the result when the ange of finvalid egister i bllows, to Sou Sou | ruction purposi- registe t is ro s source $2^{31} - 2^{31}$ opera- s set as accord rce is a rce is a | a arithma e registe r reg3. unded ir e operat to -2^{31} tion exc s an inva ing to d a positiv a negativ | etical r reg n the ad is l, an 1 septic alid o iffere re nun ve nu | 1 y = 2 t c $-\infty$ infi IEF perconsecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecution mboxecuti | conv o 32- direct inite EE75 are r ration es an er or oer, n | Preference bit corr 4-d not not non $+\infty$ oot-a | the fixed fixed n, rej not-a efine enab nd no g sou g sou x: 2 ³¹ | sin l-p gar -nu ed i led o ex urc - mbe | gle oint dle umb nva , th ccep es. 1 is er, c | -pi t fc ss e r otic re | recissormation of the formation of the | sion at, a me c whe erat erv ccu med. -2^3 | flo und urr en t tior tio rs. | rent : he ro n bi The s ret | ng-po res th round cound coptic t (bit retur | int for e result ed result on is d 4) of t en valu | mat It in node ilt is etec the ie d | con gen e. sout teted. FPSJ | tents eral- side R s as | s of |
| [Floating-point operation exceptions] | U] Iı Iı | nimple valid o nexact e | mente peratio xcepti | d operation exception (I) | ion e otion | xce (V) | eption) | n (E | E) | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | +Normal | –Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|---------|---------|--------|-------|-----------------|----|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | +Max Int [V] | | -Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| FMAF.S | | | | | | | | | | | | | | | Flo | atin | g-pc | oint | Fus | sed-M | ultipl | ly-ac | Id (S | Sing | le) |
|-----------------------------------------|-----|------------------------------------------------------|---------------------------------------------|---------------------------------------------------|-----------------------------------------------------------|--------------------------------------------------|-----------------------------------------------------|----------------------------------|----------------------------------------------------------------|-----------------------------------------|-------------------------------------|------------------------|---------------------------------------------|----------------------------------------|----------------------------------------------|------------------------------------------|-------------------------------------|------------------------------------------|------------------------------------|--------------------------------------------------------|------------------------------------|-----------------------------------------|-----------------------------------------------|-------------------------------|------------------|
| | | | | | | | | | | Flo | ati | ng- | -poir | nt fus | ed-n | nulti | ply-a | add | ope | eratio | ו (sir | ngle | pre | cisic | on) |
| [Instruction format] | | FMA | F.S | reg1, | reg2, | reg3 | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ∢ | — fi | ma (re | g2, re | g1, r | reg3) |) | | | | | | | | | | | | | | | | | |
| [Format] | | Forma | at F | :I | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 44 | 10 | | | F | 4 | | | , <i>·</i> | 04 | | | 07 | | 25 | | 22 | 22.0 | 1 00 | | | 47 | 16 |
| | r r | r r | r | 1 1 | 1 1 | 1 | 1 I | 4 R | R R | R I | 2 | w | w | w v | v w | 1 | 0 | 0 | 1 | 1 1 | . 0 | 0 | 0 | 0 | 0 |
| | | reg2 | | | | | | | reg1 | | | | r | eg3 | | | cat | tego | ory | type | | sub | -op | | |
| [Description] | | This i purpo purpo purpo execu but th | nstr se r se r se r ted e re | egiste egiste egiste as if it sult of | n mult r reg2 r reg1 r reg3 t were f the a | iplie with , adc , anc of in dd o | s the n the ls the l stor nfini pera | e si e si res te tic | ingle-p ingle- single- s the re accura on is re | preci preci prec esult acy. | sic sic isi in Th ed | on on ge ie i | floa floa flo ener resu n ac | ting ating al-p lt of cord | g-poi -poi g-po urpo the lanc | int f nt f int ose mu e w | forn forn regi ltip ith | nat nat nat iste ly c the | cor cor t co er re ope | ntents ntents ntent eg3. T ration rrent | s in s in the n is rou | gen gen gen ope not ndir | eral eral iera ratio roun ng n | - 1- on i nde nod | is :d, le. |
| [Floating-point operation exceptions |] | Unim Invali Inexa Overf Under | pleı d oj ct e low | nenteo peratic xcepti y excep w exce | d oper on exc on (I) otion (eption | ation eptic (O) (U) | n exc | cep √) | otion (| E) | | | | | | | | | | | | | | | |



RH850G3MH Software

[Operation result]

| | reg2(B) | | | | | | | | |
|------------|------------|----------|----------|----------|------------|----------|----------|-------|------------|
| reg3(C) | reg1(A) | + Normal | – Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
| | +Normal | | | • | | +∞ | _∞ | | |
| | -Normal | | FMA (/ | A, B, C) | | _∞ | +∞ | | |
| ±Normal | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | _∞ | | | +∞ | _∞ | | |
| | _∞ | -∞ | +∞ | Q-116 | antv | _∞ | +∞ | | |
| | +Normal | | | | | +∞ | -∞ | | |
| | -Normal | | FMA (/ | A, B, C) | | -∞ | +∞ | | |
| ±0 | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | _∞ | | | +∞ | _∞ | | |
| | _∞ | _∞ | +∞ | Q-No | antv | _∞ | +∞ | | |
| | +Normal | | | • | | +∞ | Q-NaN[V] | | |
| | -Normal | | + | -00 | | Q-NaN[V] | +∞ | | |
| +∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | Q-NaN[V] | | - N I() /1 | +∞ | Q-NaN[V] | | |
| | _∞ | Q-NaN[V] | +∞ | Q-Na | an[v] | Q-NaN[V] | +∞ | | |
| | +Normal | | | • | | Q-NaN[V] | _∞ | | |
| | -Normal | | - | -∞ | | _∞ | Q-NaN[V] | | |
| _∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | Q-NaN[V] | _∞ | | - N I() /1 | Q-NaN[V] | _∞ | | |
| | _∞ | _∞ | Q-NaN[V] | Q-Na | an[v] | _∞ | Q-NaN[V] | | |
| | ±Normal | | | • | | • | | | |
| Q-NaN | ±0 | | | Q-1 | NaN | | | | |
| | ±∞ | | | | | | | | |
| Not S-NaN | Q-NaN | | | | | | | Q-NaN | |
| Don't care | S-NaN | | | | | | | | |
| S-NaN | Don't care | | | | | | | | ע-ואמוא[۷] |

Note 1. [] indicates an exception that must occur.

Note 2. When the FS bit of the FPSR register is 1, subnormal numbers are flushed to the normalized numbers shown in **Section 6.1.8**, **Flushing Subnormal Numbers**.

[Supplement]

The operation is executed as if it were of infinite accuracy and the result is rounded in accordance with the current rounding mode. The result therefore differs from the result obtained when using a combination of the ADDF and MULF instructions.



.

| FMSF.S | | | | | | | | | F | loati | na-i | poir | nt fu | FIC | nulti | g-po plv-s | unt H | ract or | -Mu | atior | y-su 1 (sir | btrac | t (Sii | ngle sior | 3) 1) |
|-----------------------------------------|-----|-------------------------------------------------------------------|------------------------------------------------------|-------------------------------------------------------|-----------------------------------------------------|-----------------------------------------------|---------------------------|----------------------------------------------|------------------------------------------------|----------------------------------------|------------------------------------------------|-----------------------------------------|-------------------------------------------|-----------------------------------------------------|----------------------------------|-------------------------------------------|---------------------------------------------|-------------------------------------------------|------------------------------------------|-------------------------------------|-------------------------------------------------|-----------------------------------------------|----------------------------------------------|-------------------------|---------------|
| [Instruction format] | | FMSF.S | reg | l, reg | g2, r | eg3 | | | | | | | | | | <u> </u> | | | | | <u> </u> | <u> </u> | | | - |
| [Operation] | | reg3 ← | fms (| reg2 | 2, reg | g1, re | eg3) |) | | | | | | | | | | | | | | | | | |
| [Format] | | Format l | F:I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | 1 | 1 10 | | | | 5 | 4 | | | 0 | 31 | | | 2 | 7 26 | 25 | 23 | 3 22 | 2 21 | 1 20 | | 1 | 71 | 6 |
| | r r | rrı | 1 | 1 1 | 1 1 | 1 | 1 | RI | R R | R | R | w | W | w v | V W | 1 | 0 | 0 1 | 1 | . 1 | 0 | 0 | 0 1 | |) |
| | | reg2 | | | | | | | reg | 1 | | | I | reg3 | | | ca | tegory | t | уре | 1 | sub- | ор | | |
| [Description] | | This inst purpose purpose executed but the r mode. | tructi regis regis regis l as i esult | on n ater r ater r ater r f it w of tl | nultij reg2 reg1, reg3, vere o he su | plies with sub and of in lbtra | the the sto fini | e sir sir ts tl res te a oper | ngle- ngle- he si the accu atio | -pre- ingle resu racy n is | cisi cisi e-pi llt ii 7. Tl rou | ion on rec: n g he : und | flo floa isio ene resu ed, | ating ating n flo ral-p ılt of in ac | -po -po atir urp the | int f int f g-p ose mu dan | form form oint reg ltip ce v | nat con t form ister i ly op with t | onte onte nat regi era he | ents ents con 3. T tior | in ; in ; iten The ; i is ; rent | gene gene ts in oper not r rou | ral- ral- gen ation ound ndin | era n is ded g | 1- 3 I, |
| [Floating-point operation exceptions |] | Unimple Invalid o Inexact o Overflov Underflo | emen opera excep w exc ow exc | ted of tion otion cepti | opera exce (I) on ((tion) | ntion eptio O) (U) | exo n (V | cept V) | tion | (E) | | | | | | | | | | | | | | | |



RH850G3MH Software

[Operation result]

| | reg2(B) | | | | | | | | |
|------------|------------|----------|----------|----------|-------|----------|------------|-------|------------|
| reg3(C) | reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| | +Normal | | | | | +∞ | -8 | | |
| | –Normal | | FMS (/ | A, B, C) | | _∞ | +∞ | | |
| ±Normal | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | _∞ | Q-Na | aN[V] | +∞ | _∞ | | |
| | -∞ | _∞ | +∞ | | | -∞ | +∞ | | |
| | +Normal | | | • | | +∞ | -8 | | |
| | -Normal | | FMS (/ | A, B, C) | | _∞ | +∞ | | |
| ±0 | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | _∞ | Q-Na | aN[V] | +∞ | _∞ | | |
| | _∞ | _∞ | +∞ | | | _∞ | +∞ | | |
| | +Normal | | | • | | Q-NaN[V] | -8 | | |
| | -Normal | | - | | | _∞ | Q-NaN[V] | | |
| +∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | Q-NaN[V] | _∞ | Q-Na | aN[V] | Q-NaN[V] | _∞ | | |
| | _∞ | _∞ | Q-NaN[V] | | | _∞ | Q-NaN[V] | | |
| | +Normal | | | • | | +∞ | Q-NaN[V] | | |
| | –Normal | | + | .∞ | | Q-NaN[V] | +∞ | | |
| _∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | Q-NaN[V] | Q-Na | aN[V] | +∞ | Q-NaN[V] | | |
| | _∞ | Q-NaN[V] | +∞ | | | Q-NaN[V] | +∞ | | |
| | ±Normal | | | • | | • | | | |
| Q-NaN | ±0 | | | Q-1 | laN | | | | |
| | ±∞ | | | | | | | | |
| Not S-NaN | Q-NaN | | | | | | | Q-NaN | |
| Don't care | S-NaN | | | | | | | | |
| S-NaN | Don't care | 1 | | | | | | | Q-ivalv[V] |

Note 1. [] indicates an exception that must occur.

Note 2. When the FS bit of the FPSR register is 1, subnormal numbers are flushed to the normalized numbers shown in **Section 6.1.8, Flushing Subnormal Numbers**.

[Supplement]

The operation is executed as if it were of infinite accuracy and the result is rounded in accordance with the current rounding mode. The result therefore differs from the result obtained when using a combination of the SUBF and MULF instructions.

| FNMAF.S | • | | | | | | Floati | ng-poi | int F | ⁻ used-Neg | ate-Mu | ltiply-add | ל (Single |) |
|-----------------------------------------|----------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|----------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------|--------------------------------------------|------------------------------------------------------|---------------------------------------------------------------------------|------------------------------------------------------------|----------------------------------------------------------------|-----------------------------------------------|---|
| | | | | | Flo | pating | g-point fu | sed-m | ulti | ply-add op | eration | (single p | precision |) |
| [Instruction format] | FNMAF | .S reg1, reg2 | 2, reg3 | | | | | | | | | | | |
| [Operation] | reg3 ← 1 | neg (fma (reg | g2, reg1 | , reg3)) | I | | | | | | | | | |
| [Format] | Format F | F:I | | | | | | | | | | | | |
| [Opcode] | 15 1 | 1 10 | F | 4 | | 0.3 | 1 | 27 | 26 | 25 22 | 22.24 | 20 | 17 1(| 2 |
| | r r r r r | · 1 1 1 1 | 1 1 | R R I | RR | R W | w w | w w | 1 | 25 23 0 0 1 | 1 1 | 0 0 | 1 0 0 | |
| | reg2 | | | re | g1 | | reg3 | | | category | type | sub- | ор | |
| [Description] | This inst purpose purpose purpose The oper operation current r | ruction mult register reg2 register reg1 register reg3 ration is exec n is not round ounding mod | iplies th with th , adds th , inverts cuted as led, but le. The | the single the single the single is the sign if it we the rest signs at | e-prec e-prec gn, and ere of i ult of t re reve | isior isior cisio 1 sto infin the a ersec | n floatin n floatin n floatir res the r ite accu dd opera l after ro | g-poi g-poi result racy. ation | nt f nt f int t in § The is re ng. | ormat co ormat co format co general-p e result o ounded, i | ntents ntents ontents urpose f the m n acco | in gene in gene in gene registe nultiply rdance | ral- ral- eral- er reg3. with the | 2 |
| [Floating-point operation exceptions | Unimple] Invalid o Inexact e Overflow Underflo | emented oper operation exc exception (I) w exception (ow exception | ation ex eption ((O) (U) | (V) | n (E) | | | | | | | | | |



RH850G3MH Software

[Operation result]

| \searrow | reg2(B) | | | | | | | | |
|------------|------------|----------|----------|-----------|-----------|----------|----------|-------|------------|
| reg3(C) | reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
| | +Normal | | | • | | _∞ | +∞ | | |
| | –Normal | | FNMA | (A, B, C) | | +∞ | _∞ | | |
| ±Normal | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | _∞ | +∞ | | | _∞ | +∞ | | |
| | _∞ | +∞ | _∞ | | antv | +∞ | _∞ | | |
| | +Normal | | | • | | _∞ | +∞ | | |
| | –Normal | | FNMA | (A, B, C) | | +∞ | _∞ | | |
| ±0 | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | _∞ | +∞ | | | _∞ | +∞ | | |
| | _∞ | +∞ | _∞ | Q-IN | antvi | +∞ | _∞ | | |
| | +Normal | | L | | | _∞ | Q-NaN[V] | | |
| | -Normal | | - | -∞ | | Q-NaN[V] | _∞ | | |
| +∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | _∞ | Q-NaN[V] | | - NI() /1 | _∞ | Q-NaN[V] | | |
| | _∞ | Q-NaN[V] | _∞ | Q-IN | antvi | Q-NaN[V] | _∞ | | |
| | +Normal | | | • | | Q-NaN[V] | +∞ | | |
| | -Normal | | + | - 00 | | +∞ | Q-NaN[V] | | |
| _∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | Q-NaN[V] | +∞ | Q-Na | aN[V] | Q-NaN[V] | +∞ | | |
| | _∞ | +∞ | Q-NaN[V] | | | +∞ | Q-NaN[V] | | |
| | ±Normal | | L | | | | L | | |
| Q-NaN | ±0 | | | Q-1 | NaN | | | | |
| | ±∞ | | | | | | | | |
| Not S-NaN | Q-NaN | | | | | | | Q-NaN | |
| Don't care | S-NaN | | | | | | | | |
| S-NaN | Don't care | 1 | | | | | | | ∿-เงลเง[۷] |

Note 1. [] indicates an exception that must occur.

Note 2. When the FS bit of the FPSR register is 1, subnormal numbers are flushed to the normalized numbers shown in **Section 6.1.8**, **Flushing Subnormal Numbers**.

[Supplement]

The operation is executed as if it were of infinite accuracy and the result is rounded in accordance with the current rounding mode. The result therefore differs from the result obtained when using a combination of the ADDF, MULF, and NEGF instructions.



| FNMSF.S |) | | | | | | | | | | | | F | loati | ing-p | oint | Fuse | ed-N | lega | te-l | Mult | iply- | -sut | otrac | t (Sin | gle) | |
|-----------------------------------------|----------|-------------------------------------------------------------|---------------------------------------------------|---------------------------------------------------|----------------------------------------------------|--------------------------------------------|-----------------------------------------------|------------------------------------|----------------------------------------------------|-------------------------------------------|--------------------------------------------------------|--------------------------------------------------------|------------------------------------------------|---------------------------------------------------|----------------------------------------------------------------------|------------------------------------------------------|----------------------------------------|--------------------------------------------|----------------------------------------------------|------------------------------------------------|---------------------------------------------|-----------------------------------------------|------------------------------------------|-------------------------------------------------|---------------------------------------|-------------|---|
| | | | | | | | | | | F | Float | ing-j | poi | nt fu | ised- | mult | iply- | subt | ract | ор | erat | ion | (sin | gle p | recis | ion) | |
| [Instruction format] | | FNMS | F.S | reg | ;1, re | eg2, | reg | 3 | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | - ne | eg (f | ms (| reg2 | 2, re | g1, | reg | (3)) | | | | | | | | | | | | | | | | | |
| [Format] | | Format | t F: | Ι | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | | 11 | 10 | | | | 5 | 4 | | | 0 | 31 | | | 2 | 7 26 | 25 | - | 23 | 22 | 21 | 20 | | 17 | <u>′ 16</u> | 1 |
| | rı | rrr | r | | | T | T | | R I | K R | K R | R | W | W | W | w v | V I | 0 | 0 | | | T | 0 | 0 | | 0 | ļ |
| | | reg2 | | | | | | | | reg | g1 | | | I | reg3 | | | ca | teg | ory | ty | ре | | sub- | эр | | |
| [Description] | | This in purpos purpos The op operati with th | e re e re e re e re e ra on e c | uctio egist egist tion is no curre | er re er re er re is ez ot ro nt ro | ultip g2 v g1, g3, kecu und | blies with sub inve ited ed, l | the trac erts as i but | e sir sin ts tl the f it the de. | ngle ngle sig wer resu The | -pre -pre ingl n, ai re of ult o sig | ecisi cisi e-pi nd s f inf f th ns a | ion on rec stor fini e s are | flo flo isic rest ite a ubt rev | eating ating on flo the r accur ract ract rerse | g-po g-pc patin esul racy ope d af | oint ng-p t in t. Th ratio | forr forr oin ger e re on i | mat nat t fo nera esul s ro ndin | con con rma 1-p t of unc ig. | nter nten at c urp f the ded | nts i nts i conte ose e m , in | in g n g ent reg ulti acc | gener gener s in giste iply cord | ral- cal- gene r reg ance | eral- 3. | |
| [Floating-point operation exceptions |] | Unimp Invalid Inexac Overfle Underf | len l op t ex ow lov | nente berati ccept exce w exce | ed op ion e tion eptio cepti | pera exce (I) on (C ion (| tion ptio)) (U) | n exa | cept √) | tion | (E) | | | | | | | | | | | | | | | | |



RH850G3MH Software

[Operation result]

| | reg2(B) | | | | | | | | |
|------------|------------|----------|----------|-----------|--------------|----------|----------|-------|------------|
| reg3(C) | reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
| | +Normal | | | | | _∞ | +∞ | | |
| | -Normal | | FNMS | (A, B, C) | | +∞ | _∞ | | |
| ±Normal | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | _∞ | +∞ | 0.01 | | _∞ | +∞ | | |
| | _∞ | +∞ | _∞ | Q-110 | מויעניין | +∞ | _∞ | | |
| | +Normal | | • | • | | _∞ | +∞ | | |
| | -Normal | | FNMS (| (A, B, C) | | +∞ | _∞ | | |
| ±0 | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | _∞ | +∞ | | - N I [\ /] | _∞ | +∞ | | |
| | _∞ | +∞ | _∞ | Q-IN | antvi | +∞ | _∞ | | |
| | +Normal | | | | | Q-NaN[V] | +∞ | | |
| | -Normal | | + | | | +∞ | Q-NaN[V] | | |
| +∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | Q-NaN[V] | +∞ | 0.14 | - N IF) /1 | Q-NaN[V] | +∞ | | |
| | _∞ | +∞ | Q-NaN[V] | Q-Na | antv] | +∞ | Q-NaN[V] | | |
| | +Normal | | | | | _∞ | Q-NaN[V] | | |
| | -Normal | | - | | | Q-NaN[V] | _∞ | | |
| _∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | _∞ | Q-NaN[V] | 0.14 | - N IF) /1 | _∞ | Q-NaN[V] | | |
| | _∞ | Q-NaN[V] | _∞ | Q-IN | anitvj | Q-NaN[V] | _∞ | | |
| | ±Normal | | | | | • | | | |
| Q-NaN | ±0 | | | Q-1 | NaN | | | | |
| | ±∞ | | | | | | | | |
| Not S-NaN | Q-NaN | | | | | | | Q-NaN | |
| Don't care | S-NaN | | | | | | | | |
| S-NaN | Don't care | 1 | | | | | | | Q-INAIN[V] |

Note 1. [] indicates an exception that must occur.

Note 2. When the FS bit of the FPSR register is 1, subnormal numbers are flushed to the normalized numbers shown in **Section 6.1.8**, **Flushing Subnormal Numbers**.

[Supplement]

The operation is executed as if it were of infinite accuracy and the result is rounded in accordance with the current rounding mode. The result therefore differs from the result obtained when using a combination of the SUBF, MULF, and NEGF instructions.



| MAXF.D | | | | | | | | | | | | | | | | | | Flo | batin | ig-p | oir | nt Ma | axir | num | ı (Do | oubl | e) |
|--------------------------------------|----|-----------------------------------------------------------------------------|-------------------------------------------------|--------------------------------------------------|--------------------------------------------|-----------------------------------|------------------------------------|------------------------------------|------------------------------------|-------------------------------------|-----------------------------------|----------------------------|----------------------------------|--------------------------------------|----------------------------------|----------------------------|------------------------------|----------------------------|--------------------------------|---------------------|--------------------|-----------------------------|-----------------------|----------------------------|------------------------|-------------------|---------|
| | | | | | | | | | | | | | | Floa | ting | J-po | oint | ma | kimu | ım ' | val | ue (| dou | ble | prec | isio | n) |
| [Instruction format] | | MAXF.D | reg1 | , reg2 | , reg | 3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← n | nax (re | eg2, ro | eg1) | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | :I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 11 | 10 | | | 5 | 1 | | | | 0 | 21 | | | | 27 | 26 | 25 | | 22 | 22 | 21 | 20 | | , | 7 | 16 |
| | r | rrr 0 | 1 1 | 1 1 | 1 | 1 | R | R | R | R | 0 | w | w | w | w | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | | reg2 | | | | | | I | reg1 | | | | I | reg3 | | | | cat | ego | ry | ty | pe | | sub | -op | | |
| [Description] | | This instr data in th the regist If one of detected. occurs. | ruction e regi er pai the so If inv | n extra ster pa r spec urce c alid o | acts f air sj ified opera pera | the pec l by and tior | ma ifie y ge s is n ex | xin ed b ener S-l kcej | num y ge al-p NaN ptio | n va ene ourj N, a ns a | alu eral pos in l are | e fr -pu se r IEE | om irpo egi: E7 t er | the ose r ster 54-c nabl | do regi reg lefi ed, | ub ste 3. ne Q | le-j ers i d ii -Na | orec reg nval N i | cisic l an lid c s st | on f id r ope | flo reg erat | atin 2, a tion and | g-I nd ex no | ooin stor cep exc | nt fo res i tion | rma t ir is | at 1 |
| [Floating-point operation exceptions |] | Invalid o | peratio | on exc | cepti | on | (V) |) | | | | | | | | | | | | | | | | | | | |
| [Supplement] | | When bo | th reg | 1 and | reg2 | 2 is | eitl | her | +0 | or - | -0, | it i | s u | nde | fine | ed ' | wh | ethe | er+(|) 01 | r -(|) is | sto | red | in r | egź | 3. |
| | | A subnor | mal ir | nput w | vill n | ot l | be f | lus | hed | ev | en | if t | he | FS ł | oit | of | the | FP | SR | reg | gist | er i | s 1 | | | | |



RH850G3MH Software

[Operation result]

| reg2(B) | | | | | | | | |
|---------|---------|---------|-------|--------|----|------------|----------|-----------|
| reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| +Normal | | | | | | | | |
| -Normal | | | | | | | | |
| +0 | | | | | | | rog1 (A) | |
| -0 | | | IVIAA | (А, Б) | | | regi (A) | |
| +∞ | | | | | | | | |
| -∞ | | | | | | | | |
| Q-NaN | | | reg2 | 2 (B) | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note: [] indicates an exception that must occur.



| MAXF.S | | | | | | | | | | | | | F | loating | -po | int N | /laxi | mum | (Sir | ngle) | |
|--------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------------|-------------------------------------------|------------------------------|-----------------------------------|-------------------------------|-------------------------|----------------------------|-------------------------|---------------------------------|------------------------------|-------------------------------|-----------------------------|----------------------------------------|--------------------------|----------------------------|-----------------------------|----------------------------------|---------------------------------|-------------------------|---|
| | | | | | | | | | | Floa | ting- | poir | nt ma | aximur | n va | alue | (sin | gle p | recis | sion) | |
| [Instruction format] | MAXF.S | reg1, reg2, | reg3 | | | | | | | | | | | | | | | | | | |
| [Operation] | reg3 ← m | ax (reg2, re | g1) | | | | | | | | | | | | | | | | | | |
| [Format] | Format F: | I | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 11 | 10 | 5 | 4 | | | 0 | 31 | | | 27 | 7 26 | 25 | 23 | 22 | 21 | 20 | | 1- | 7 16 | |
| | r r r r r | 1 1 1 1 | 1 1 | R | R R | R | R | w | W | w v | 7 W | 1 | 0 | 0 0 | 1 | 1 | 0 | 1 (|) O | 0 | I |
| | reg2 | | | | reg | 1 | | | r | eg3 | | | ca | tegory | ty | /pe | | sub-c | р | | |
| [Description] | This instr data in ge If one of t detected. occurs. | uction extrac neral-purpos he source oj If invalid op | cts the se regis perand peration | max sters s is n ex | ximur s reg1 S-Na ceptic | m va and N, ai ons a | lue re n II re | e fro g2, EEI not | om an E75 : en | the s d sto 54-de able | sing res efine d, Q | le-p it ir ed i 2-Na | orec n ge nva aN i | ision f neral- lid op is stor | floa pur era ed | ting pos tion and | g-po le re l ex no | oint f giste cepti exce | forn er re ion : eptic | nat eg3. is on | |
| [Floating-point operation exceptions | Invalid op 3] | peration exce | eption | (V) | | | | | | | | | | | | | | | | | |
| [Supplement] | When bot A subnorr | h reg1 and r nal input wi | eg2 is ll not l | eith be f | er +0 lushee | or – d eve | 0, i en i | it is if th | s ur ne H | ndefi FS bi | ned t of | wh the | ethe FP | er +0 SR re | or - gis | -0 is ter i | s ste is 1 | ored | in r | eg3. | |

[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|-------|--------|------------|---|----------|-----------|
| reg1(A) | Normal | –Normal | +0 | -0 | + ® | 8 | Q-NaN | S-NaN |
| +Normal | | | | | | | | |
| -Normal | | | | | | | | |
| +0 | | | MAY | | | | rog1 (A) | |
| -0 | | | IVIAA | (А, D) | | | iegi (A) | |
| +∞ | | | | | | | | |
| _∞ | | | | | | | | |
| Q-NaN | | | reg2 | 2 (A) | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note: [] indicates an exception that must occur.



| MINF.D | | | | | | | | | | | | | | | Flo | oating | I-poi | int N | /linir | num | (Do | uble | ¥) |
|------------------------------------------|---------------------------------------------------------|---------------------------------------|----------------------------------------------------|--------------------------------------------------|---------------------------------------|----------------------------------------|---------------------------------------------|---------------------------------------|------------------------------------|-----------------------------------|----------------------------------|-------------------------------------------|---------------------------------------|-----------------------------|--------------------------------|---------------------------------|--------------------|------------------------------|--------------------|----------------------------|-------|-------------------------|----|
| | | | | | | | | | | | | Float | ing-p | oin | t mir | imum | i val | ue (| dou | ble | oreci | sior | 1) |
| [Instruction format] | MINI | .D | reg1, | reg2, | reg3 | | | | | | | | | | | | | | | | | | |
| [Operation] | reg3 · | — m | nin (re | g2, re | g1) | | | | | | | | | | | | | | | | | | |
| [Format] | Form | at F: | I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | 11 | 10 | | 5 | 4 | | | 0 | 31 | | | 27 | 26 | 25 | 23 | 3 22 | 21 | 20 | | 1 | 7 1 | 6 |
| | rrrr | 0 | 1 1 | 1 1 | 1 1 | R | RR | R | 0 | W | W | W W | r 0 | 1 | 0 | 0 0 | 1 | 1 | 1 | 1 | 0 1 | L C |) |
| | reg2 | | | | | | reg | g1 | | | r | eg3 | | | cat | egory | ty | /pe | | sub- | ор | | |
| [Description] | This i data i the re If one detect occur | nstru giste of t ed. 1 s. | uction e regis er pair the sou If inva | a extra ster pa speci urce o alid oj | it spe fied b peran- peratio | e mi cifie y ge ds is on e | nimu ed by eneral s S-Na xcepti | m va gene -pur aN, a ions | ilue eral pos an I are | e fro -pu se ro EE no | om rpo egis E75 t en | the d se re ter re 54-de able | oub giste eg3. efine d, Q | le-p ers ed in -Na | oreci reg1 nval nN ia | ision and id op s stor | floa reg era | atin ;2, a tion and | g-p and n ex | oint stor cep exc | tion | mat t in is on | t |
| [Floating-point operation exceptions] | Invali] | d op | peratio | on exc | eptior | (V |) | | | | | | | | | | | | | | | | |
| [Supplement] | When | bot | h reg l | l and : | reg2 is | s eit | her +(|) or | -0, | wh | neth | er +(|) or | -0 | is st | ored | in r | eg3 | is | und | efin | ed. | |
| | A sub | nori | nal in | put w | ill not | be | flushe | ed ev | en | if t | he I | FS bi | t of | the | FPS | SR re | gis | ter i | is 1 | | | | |



RH850G3MH Software

[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|----------|--------|----|------------|----------|-----------|
| reg1(A) | Normal | -Normal | +0 | -0 | +∞ | <u>_</u> ∞ | Q-NaN | S-NaN |
| +Normal | | | | | | | | |
| -Normal | | | | | | | | |
| +0 | | | | | | | rog1 (A) | |
| -0 | | | IVIIIN (| (А, Б) | | | regi (A) | |
| +∞ | | | | | | | | |
| _∞ | | | | | | | | |
| Q-NaN | | | reg2 | 2 (B) | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note: [] indicates an exception that must occur.



| MINF.S | | | | | | | | | | | | | | | Flo | atin | a-pa | oint | Flo | pating | g-po n va | oint alue | Min (sir | imu nale | m (s | Sing | le) on) |
|-----------------------------------------|-----------|-------------------------------------------------|---------------------------------|--------------------------------------|---------------------------------|-----------------------------|---------------------------------|-----------------------------|-------------------------------|----------------------------|-----------------------------|---------------------------|---------------------------|---------------------------|--------------------------------|-------------------------------|----------------------------|---------------------------|-------------------------------|--------------------------------|-------------------------|-----------------------------|------------------------------|---------------------------|------------------------------|---------------------------|--------------|
| [Instruction format] | | MINF | S.S | reg1, | reg2 | , reg | g3 | | | | | | | | | | <u> </u> | | | | | | | <u> </u> | | | , |
| [Operation] | | reg3 ∢ | — n | nin (re | eg2, 1 | reg1 |) | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Forma | ıt F | :I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | 10 | | | - | | | | | 0 | 0.4 | | | | - 0 | 0.0 | - | 00 | | | 00 | | | 47 | 40 |
| | 15 r r | r r r | 11 r | 10 | . 1 | 1 | 5 1 1 | 4 R | R | R | R | R | 31 w | W | W | w v | 7 2 v 1 | | . 5) (|) 0 | 1 | 1 | 0 | 1 | 0 | 17 | 16 0 |
| | | reg2 | | | | | | | r | eg1 | | | | r | eg3 | | | 0 | cate | gory | ty | /pe | | sub | o-op | | |
| [Description] | | This i data in If one detect occurs | nstr n ge of ed. s. | uction neral- the so If inv | n ext -purp ource alid | racts pose ope ope | s the regi erand ratio | mi stei ls is n ez | nim rs re s S-N kcep | um g1 a NaN otior | val and [, ai ns a | lue l re n I are | e fro eg2, EE no | om , an E7: t er | the a d sto 54-d able | sing ores efin ed, 9 | it i it i led Q-N | pre in g inv NaN | cisi gene valio V is | ion f eral- d op stor | loa pui era ed | ting rpos tior and | g-po se r n ex l nc | oint egis cer ex | t foi ster otio cep | ma reg n is tion | t 3. 1 |
| [Floating-point operation exceptions |] | Invali | d oj | perati | on ex | ксер | otion | (V) |) | | | | | | | | | | | | | | | | | | |
| [Supplement] | | When A sub | bot nor | th reg mal iı | 1 and | d reg will | g2 is not | eitl be f | her · flusl | +0 d | or – eve | -0, en | wh if t | ieth | er + FS b | 0 o it o | r –(f th |) is le F | sto PS | red R re | in 1 gis | reg3 ter | 3 is is 1 | uno | defi | ned | • |



RH850G3MH Software

[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|--------|--------|------------|---|----------|-----------|
| reg1(A) | Normal | –Normal | +0 | -0 | + ∞ | 8 | Q-NaN | S-NaN |
| +Normal | | | | | | | | |
| -Normal | | | | | | | | |
| +0 | | | | | | | rog1 (A) | |
| -0 | | | IVIIIN | (А, Б) | | | regi (A) | |
| +∞ | | | | | | | | |
| _∞ | | | | | | | | |
| Q-NaN | | | reg2 | 2 (B) | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.

[Operation result]

| | reg2(B) | | | | | | | | |
|------------|------------|----------|----------|-----------|----------|----------|------------|-------|-------|
| reg3(C) | reg1(A) | +Normal | –Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| X | +Normal | | | | | +∞ | _∞ | | |
| | –Normal | | MUSB (| (A, B, C) | | _∞ | +∞ | | |
| ±Normal | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | _∞ | | | +∞ | _∞ | | |
| | _∞ | ∞ | +∞ | Q-IN | antvi | ∞ | +∞ | | |
| | +Normal | | | | | +∞ | 8 | | |
| | –Normal | | MUSB (| (A, B, C) | | | +∞ | | |
| ±0 | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | -∞ | | | +∞ | _∞ | | |
| | _∞ | | +∞ | Q-ING | antvi | | +∞ | | |
| | +Normal | | | | | Q-NaN[V] | -8 | | |
| | –Normal | | - | .∞ | | | Q-NaN[V] | | |
| +∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | Q-NaN[V] | _∞ | | N I() /1 | Q-NaN[V] | _∞ | | |
| | _∞ | | Q-NaN[V] | | antvi | | Q-NaN[V] | | |
| | +Normal | | | | | +∞ | Q-NaN[V] | | |
| | –Normal | | + | .∞ | | Q-NaN[V] | +∞ | | |
| _ ∞ | ±0 | | | | | Q-Na | aN[V] | | |
| | +∞ | +∞ | Q-NaN[V] | | | +∞ | Q-NaN[V] | | |
| | _∞ | Q-NaN[V] | +∞ | Q-ING | antvi | Q-NaN[V] | +∞ | | |
| | ±Normal | | | | | | | | |
| Q-NaN | ±0 | | | Q-1 | laN | | | | |
| | ±∞ | | | | | | | | |
| Not S-NaN | Q-NaN | | | | | | | Q-NaN | |
| Don't care | S-NaN | | | | | | | | |
| S-NaN | Don't care | | | | | | | | |

Note 1. [] indicates an exception that must occur.

Note 2. When the FS bit of the FPSR register is 1, subnormal numbers are flushed to the normalized numbers shown in **Section 6.1.8, Flushing Subnormal Numbers**.

RENESAS

[Description]

The operation is executed as if it were of infinite accuracy and the result is rounded in accordance with the current rounding mode. The result therefore differs from the result obtained when using a combination of the SUBF and MULF instructions.



| MULF.D | | | | | | | | | | | | | | | F | loatii | ng-poin | t Multi | ply (E | oubl | e) |
|-----------------------------------------|-----------|---------------------------------------------------------------|-----------------------------------------------------|-------------------------------------------------|-----------------------------------------|--------------------------|---------------------------|---------------------------|-------------------|--------------------|----------------------|----------------------|-------------------------|----------------------|------------------------|---------------------|-------------------------------|------------------------------|--------------------------|----------------------|----------|
| | | | | | | | | | | | | Flo | ating | g-po | int m | ultipl | ication | (doub | le pre | ecisio | n) |
| [Instruction format] | | MULF.D | reg1, | reg2, 1 | reg3 | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← re | eg2 × r | eg1 | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | I | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | 10 | | - | | | | 0 | 0.4 | | | 07 | ~~~ | 05 | 0 | | 00 | | 47.4 | |
| | 15 r 1 | 11 rr0 | 10 | 1 1 | 1 1 | 4 R | R F | R R | 0 | 31 w | W | w w | 7 0 | 1 | 0 | 0 0 | 1 1 | 1 | 0 1 | 0 | 0 |
| | | reg2 | | | | | re | g1 | | | re | eg3 | | | cate | gory | type | s | ub-op | | |
| [Description] | | This instr specified contents o general-p | uction by gen of the re urpose | multip eral-p egister regist | plies d urpose r pair s er reg | out e re spe 3. | ole-pr giste cified | recisi r reg l by g | ion 2 b gen | flo y tł era | atin ne d 1-pu | g-pc oubl rpos | oint f e-pr se re | forn eci: gist | nat c sion er re | onte floa g1, | ents of ting-po and sto | the re bint fo bres tl | egiste orma ne ree | er pa t sult i | ir in |
| [Floating-point operation exceptions |] | Unimpler Invalid op Inexact ex Overflow Underflow | nented peration kceptio except w except | opera n exce on (I) tion (C ption (| tion e ption D) (U) | xce (V) | ption) | ı (E) | | | | | | | | | | | | | |

[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|------|-------|-------|------------|-------|-----------|
| reg1(A) | Normal | -Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| +Normal | | | | | +∞ | -∞ | | |
| -Normal | | Δ. | . D | | | +∞ | | |
| +0 | | A 2 | КD | | | | | |
| -0 | | | | | Q-INC | מאנען | | |
| +∞ | +∞ | _∞ | Q-Na | N [V] | +∞ | _∞ | | |
| _∞ | -8 | +∞ | | | | +∞ | | |
| Q-NaN | | | | | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.

| MULF.S | | | | | | | | | | | | | | | | Float | ing- | -poin | t Mu | ltipl | y (S | ingle |
|-----------------------------------------|-----------|-----------------------------------------------------------|-----------------------------------------------|----------------------------------------------------|--------------------------------|------------|--------------------------------|----------------------------|-------------------|---------------------|-------------------------|---------------------|----------------------|----------------------|---------------------|---------------------------|----------------------|---------------------|-------------|-------|-------|------------|
| | | | | | | | | | | | | FI | oatin | g-p | oint r | nultip | olica | ation | (sing | gle | prec | isior |
| [Instruction format] | | MULF.S | reg1 | , reg2, 1 | eg3 | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← 1 | eg2 × | reg1 | | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | 7:I | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | _ | | | | _ | ~ / | | | ~- | | | | | | | | | |
| | 15 r : | 11 r r r r | 10 | 1 1 1 | 1 1 | 4 R | RR | R I | 0 R | 31 w | w v | w w | 27 7 w | 1 | 25 0 | 0 0 | 1 | <u>2 21</u> . 1 | 0 | 0 | 1 | 7 1 0 C |
| | | reg2 | | | | | reg | 1 | | | re | g3 | | | cate | egory | ţ | уре | s | sub | -ор | |
| [Description] | | This inst purpose purpose | ructio regist regist | n multij er reg2⊺ er reg1, | plies the by the and st | he sir | single- ngle-pr es the r | -preci recisi result | isi on : ir | on 1 fle 1 ge | float oatin enera | ting g-p al-p | -poi oint urpo | nt f for ose 1 | orm mat regis | at cont cont ster 1 | onte cent reg. | ents ts of 3. | of g gen | ene | eral- | |
| [Floating-point operation exceptions | ;] | Unimple Invalid o Inexact e Overflov Underflo | mente perati except v exce w exce | ed operation exce tion (I) eption (Corption) | ttion e eption O) (U) | xce (V) | eption) | (E) | | | | | | | | | | | | | | |

[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|------|---------|------|--------|-------|-----------|
| reg1(A) | Normal | –Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
| +Normal | | | | | +∞ | -∞ | | |
| -Normal | | Δ. | , D | | -8 | +∞ | | |
| +0 | | Α, | | | | | | |
| -0 | | | | | Q-No | an [v] | | |
| +∞ | +∞ | -∞ | | | +∞ | -∞ | | |
| _∞ | _∞ | +∞ | Q-Nd | מא ניין | -8 | +∞ | | |
| Q-NaN | | | | | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.

| NEGF.D | | | | | | | | | | | | | | | | | | | | | Flo | atin | ıg-p | oint | Ne | gate | ə (Do | oub | le) |
|--------------------------------------|----|-----------------------|-------------------------|----------------------------|-------------------------|---------------------|-------------|-------------|-------------|--------------|--------------|-------------|-------------|------------|-------------|-------------|-------------|-------------|------------|------|------------|-------------|--------------|---------------|--------------|------------|---------------|-------|-----|
| | | | | | | | | | | | | | | | | F | loa | ting- | poi | nt s | ign | inv | ersi | on (| dou | ble | prec | cisic | on) |
| [Instruction format] | | NEC | F.D | reg2 | l, reg | g3 | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ← n | ieg re | g2 | | | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Forn | nat F | :I | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | | 5 | 4 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 | 21 | 20 | | | 17 | 16 |
| | r | r r | r 0 | 1 | 1 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | w | w | w | w | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| | | reg2 | | | | | | | | | | | | | 1 | reg | 3 | | | cat | teg | ory | ty | pe | : | sub |)-op | | |
| [Description] | | This regis purp | instr ter p ose 1 | ructio air sp regist | on in pecif er re | vert fied g3. | ts th by | ie s gei | ign nera | n of al-p | f do purj | oubl pos | e-p e re | rec gis | isio ter | on f reg | floa g2, | ting and | g-p sto | oint | t fo th | orm e re | at c esul | cont lt ir | tent 1 ge | s o nei | f the ral- | e | |
| [Floating-point operation exceptions |] | None | e | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Supplement] | | A su | bnor | mal i | nput | t wi | 11 no | ot ł | be f | flus | shee | d ev | ven | if t | he | FS | bit | of | the | FP | SR | re | gist | er i | s 1. | | | | |



| NEGF.S | | | | | | | | | | | | | | | | F | loat | ing | -ро | int s | Floa gn i | atir nv | ng-poir ersion | nt Ne | egat gle | te (S prec | ingl isio | e) n) |
|--------------------------------------|-----|-------------------|-------------|----------------|---------------|---------------|----------------|--------------|-------------|-------------|------------|---------------|-------------|-------------|--------------|--------------|--------------|------------|------------|--------------|--------------|------------|-------------------|------------|-------------|---------------|--------------|----------|
| [Instruction format] | | NEGF | .S | reg2, | , reg | 3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | – n | eg re | g2 | | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Forma | t F | :I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | | 11 | 10 | | | | 5 | 4 | | | | 0 | 21 | | | | 7 | 26 | 25 | 2 | 2 | 22.21 | 20 | | | 17 4 | 16 |
| | r r | r r r | r | 1 1 | 1 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | w | W | W | w | w | 1 | 0 | 0 0 | 0 | 1 0 | 0 | 1 | 0 | 0 | 0 |
| | | reg2 | | | | | | | | | | | | | I | reg3 | 3 | | | cate | egor | у | type | | sub | -op | | |
| [Description] | | This ir genera | ıstr l-p | uctio urpos | n in se re | vert egist | ts th ter 1 | ne s regí | ign 2, a | n of and | the sto | e sin ores | ngle the | e-p e re | reci sult | isio t in | n fle gen | oat era | ing 1-p | g-po ourp | int f ose | foi re | rmat c gister | ont reg | ent 3. | s of | | |
| [Floating-point operation exceptions |] | None | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Supplement] | | A subi | nor | mal i | npu | t wi | ll n | ot ł | be f | flus | hec | l ev | en | if t | he] | FS | bit o | of t | he | FPS | SR r | eg | gister i | is 1 | | | | |



| RECIPF.D | Reciprocal of a Floating-point Value (Double) |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | Reciprocal (double precision) |
| [Instruction format] | RECIPF.D reg2, reg3 |
| [Operation] | $reg3 \leftarrow 1 \div reg2$ |
| [Format] | Format F:I |
| [Opcode] | 15 11 10 5 4 0 31 27 26 25 23 22 21 20 17 16 |
| | rrr0 1 1 1 1 1 1 0 0 0 0 1 w w w w 0 1 0 0 0 1 0 1 |
| | reg2 reg3 category type sub-op |
| [Description] | This instruction approximates the reciprocal of the double-precision floating-point format contents of the register pair specified by general-purpose register reg2, and stores the result in the register pair specified by general-purpose register reg3. The result differs from the result obtained by using the DIVF instruction. |
| [Floating-point operation exceptions] | Unimplemented operation exception (E) Invalid operation exception (V) Inexact exception (I) Division by zero exception (Z) Underflow exception (U) |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|-------------|---------|--------|--------|----|----|-------|--------------|
| Operation result [exception] | 1/ <i>A</i> | N [I] | –∞ [Z] | –∞ [Z] | +0 | -0 | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| RECIPF.S | 5 | | | | | | | | | | | | | Reci | proc | al of | a Flo | ating-p | oint \ | √alu | e (Sir | ngle) |
|-----------------------------------------|----|-----------------------------------------------------------|--------------------------------------------|--------------------------------------------------|----------------------------------------|---------------------------------|---------------------|--------------------------|-----------------------|-----------------------|---------------------|--------------|--------------------|-----------------|----------------------|----------------------|---------------------------|-----------------------------|---------------------|----------------|----------------|-------|
| | | | | | | | | | | | | | | | | | Re | ciproca | l (sin | igle | precis | sion) |
| [Instruction format] | | RECIPF | S re | g2, re | eg3 | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | 1 ÷ re | eg2 | | | | | | | | | | | | | | | | | | |
| [Format] | | Format l | F:I | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | 1 | 1 10 | 1 1 | 1 | 1 1 | 4 | 0 | 0 0 | 0 | 31 | | | 2 | 7 26 | 25 | 23 | 3 22 2' | 20 | 1 | 1 | 7 16 |
| | | reg2 | | <u> </u> | 1 | | | 0 | 0 0 | 1 | w | re | w eg3 | w w | | cat | egory | type | | ⊥ sub· | -op | |
| [Description] | | This inst contents reg3. Th | truction of ge the rest | on ap eneral ult dif | prox -pur fers | imate pose from | es tl reg the | he re fister e res | ecipr reg ult o | ocal 2, ai btai | l of nd s nec | the store | sin s tl usi | igle-j he re | prec sult ne E | isio in g DIVI | n floa gener 7 inst | ating-p al-pur ructio | ooint pose n. | t for e reg | rmat gister | |
| [Floating-point operation exceptions |] | Unimple Invalid o Inexact o Division Underflo | ement opera excep i by z ow ex | ted op tion e otion (ero ex ception | oerati xcep (I) xcep on (U | ion ex otion tion (U) | xce (V) Z) | ptio | n (E) | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | –Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
|------------------------------------|--------|---------|--------|--------|----|----|-------|--------------|
| Operation result [exception] | 1/A | N [I] | +∞ [Z] | –∞ [Z] | +0 | -0 | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| RSQRTE | D | | | | | F | Recip | roc | al c | of th | e Sc | qua | re F | 200 | t of a | a Fl | oat | ting- | poir | nt \ | /alue | e (Do | bub | le) |
|-----------------------------------------|---------------------------------------------------------------|----------------------------------------------------|---------------------------------------------------------|----------------------------------------------|---------------------------------------|----------------------------|-------------------------|------------------------|---------------------|-------------------|-------------------|-----------------------------|---------------------------|--------------------------------------|--------------------------|-----------------------|------------------|-----------------------|----------------------|---------------------|--------------------------------|-----------------------|-------------------|---------|
| | | | | | | | | | | | | Re | ecip | roc | al of | squ | Jar | e ro | ot (| dou | ıble | prec | isic | on) |
| [Instruction format] | RSQRTI | F.D reg | 2, reg3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | reg3 ← 1 | l ÷ (sqr | t reg2) | | | | | | | | | | | | | | | | | | | | | |
| [Format] | Format I | 7:I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 4 | | | _ | 4 | | | ~ | 0.4 | | | | 07 | ~~ | 05 | | ~~ | | | ~~~ | | | | 40 |
| | 15 1° | 1 10 | 1 1 | 1 1 | 4 | 0 0 | 1 | 0 | 31 | 7.7 | 1.7 | 7.7 | 27 | 26 1 | 25 0 | 0 | <u>23</u> | 1 | 21 | 1 | 1 | 1 | 1 | 16 |
| | I I I I U | | 1 1 | ΙI | 0 | 0 0 | T | 0 | w | w | w | w | 0 | T | 0 | 0 | 0 | 1 | 0 | T | T | 1 · | 1 | 0 |
| [Description] | This inst point for approxin general-j The resu | ruction mat connates th purpose It differ | obtains ntents o e recip: e registe rs from | s the of the rocal er reg the re | arith regi of tl 3. esult | imetio ster p his re | c pos pair s sult | siti spe an w | ive ecif d st | squ ied ore | are by s th | roo gen le ro a co | ot o nera esu om | of ti al- <u>r</u> lt i bin | ne courp n th atic | loul oose le re | ble re egi | e-pr egis ister | ecis ster r pa | sio reg tir : | n flo g2, t spec F ar | oatin hen sifie | ng- d b DIV | y YF |
| [Floating-point operation exceptions | instructio Unimple] Invalid o Inexact o Division | mented peratio exception by zero | l operat n excep on (I) o excep | ion e otion tion (| xcer (V) (Z) | otion | (E) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|-----------|--------------|-------|-------|----|--------------|-------|--------------|
| Operation result [exception] | 1/√-A [I] | Q-NaN [V] | +∞[Z] | –∞[Z] | +0 | Q-NaN [V] | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| RSQRTF | S | | | | | | | Recip | roca | al of | the | Squai | e Ro | oot o | of a Fl | oa | ting-pc | oint Va | alue (| Sin | gle) |
|-----------------------------------------|----|-----------------------------------------------------|-----------------------------------------|---------------------------------------|----------------------------------------|----------------------------|----------------------------------------|--------------------------------------|-----------------------------|----------------------------|-------------------------------|-----------------------------------|------------------------------|-----------------------------|-------------------------------------|-----------------|-----------------------------------|-------------------------|------------------------|--------------------|------|
| | | | | | | | | | | | | Re | cipro | ocal | of sq | Jai | re root | (sing | le pre | ecisi | on) |
| [Instruction format] | | RSQRTF. | S reg | 2, reg | 3 | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← 1 | ÷ (sqr | t reg2 | 2) | | | | | | | | | | | | | | | | |
| [Format] | | Format F: | Ι | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 | 11 | 10 | | 5 | 4 | | ſ |) 3' | 1 | | 2 | 7 26 | \$ 25 | | a | 22 21 | 20 | | 17 | 16 |
| | r | rrrr | 1 1 | 1 1 | 1 1 | 0 | 0 0 |) 1 (|) w | / v | v w | w | v 1 | 0 | 0 (|) | 1 0 | 0 3 | L 1 | 1 | 0 |
| | | reg2 | | | | | | | | | reg | 3 | | Са | ategor | y | type | S | np-ob |) | |
| [Description] | | This instr point form this result obtained | uction hat con and st when u | obtai ntents tores i using | ns the of ger it in ge a coml | ari nera ener oin | thmeti al-purj ral-pu ation o | ic posi pose r rpose of the | itive egis regi SQ | e so stei iste RT | quar reg er reg F an | e roo 2, the g3.] nd DI | t of en aj The : VF | the opro resu inst | singl oxima ılt dif ructio | e- ate fe | precis es the rs froi s. | ion f recip m the | loati roca e res | ng- l of ult | |
| [Floating-point operation exceptions | 5] | Unimplen Invalid op Inexact ex Division b | nented beratio cceptic by zero | l opera n exce on (I) o exce | ation e eption eption | (V (Z) | eption) | (E) | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|-----------|--------------|----------|-------|----|--------------|-------|--------------|
| Operation result [exception] | 1/√-A [I] | Q-NaN [V] | +∞ [Z] | –∞[Z] | +0 | Q-NaN [V] | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| SQRTF.D | | | | | | | | | | | | | | | | F | loati | ng-l | ooi | nt So | qua | ire F | Root | (Do | bubl | e) |
|--------------------------------------|-----|--------------------------------------------------------------------|---------------------------------------------------|------------------------------------------------|-------------------------------------------|---------------------------------------|-----------------------------------|--------------------|------------------------------------|---------------------------|-------------------------------------|-----------------------------------|--------------------------------|-------------------------------------|--------------------------------------|-----------------------------------|--------------------------------------|-------------------------------------|---------------------------------|-------------------------------------|---------------------------|----------------------------|--------------------------------------|------------------------------|-------------|--------|
| | | | | | | | | | | | | | | | | | | Squ | Jar | e roo | ot (| doul | ole p | orec | isio | n) |
| [Instruction format] | | SQRTF.D | reg2 | , reg3 | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← so | qrt reg | 2 | | | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | 11 | 10 | | | 54 | | | | 0 | 31 | | | | 27 | 26 | 25 | | 23 | 22 2 | 21 | 20 | | 1 | 7 <i>^</i> | 16 |
| | r i | rr 0 | 1 1 | 1 1 | 1 | 1 0 | 0 | 0 | 0 | 0 | w | W | w | w | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| | | reg2 | | | | | | | | | | I | reg3 | } | | | cat | ego | ry | typ | е | 5 | sub- | ор | | |
| [Description] | | This instr point form stores the is execute current ro | uction nat con result ed as if unding | obtain ntents in the f it we g mod | ns the of the regi re of e. W | e ari le re ster infi hen | thr gist pai nite the | r sp ace sou | c po pair pecificura urce | spe fiec acy, op | ive ecif l by , an oera | squ ïed / ge d tl und | by by ner ne r val | e roo ger al-p esu ue i | ot c nera ourj lt is s – | of t al-1 pos s rc 0, | he o purp se re pune the | loul oose egis ded resu | ble e re ter in ult | e-pre egist reg acc bec | ecis ter 33. orc | sion reg The lane | 1 flc 2, a e op ce v –0. | oatin ind oera vith | ng- tion | n e |
| [Floating-point operation exceptions | 5] | Unimpler Invalid op Inexact ex | nented peratio cceptic | l opera n exce on (I) | ation eptio | exc n (V | epti ') | on | (E) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | + ∞ | _∞ | Q-NaN | S-NaN |
|------------------------------------|--------|--------------|----|----|------------|--------------|-------|--------------|
| Operation result [exception] | √A | Q-NaN [V] | +0 | -0 | +∞ | Q-NaN [V] | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| SQRTF.S | | | | | | | | | | | | | | | | | F | oatir | ıg-po | oint S | qua | re Ro | oot (S | Singl | le) |
|-----------------------------------------|-----------|----------------------------------------------------------|---------------------------|-------------------------------------------------|------------------------------------------------|-----------------------------------|-----------------------------|-------------------------------|------------------------------|-------------------------------|---------------------------|-----------------------------|-----------------------------|-----------------------------|------------------------------|--------------------------|------------------------------|--------------------------------|----------------------|-----------------------------------|-------------------------------|---------------------------------|-----------------------------|-------------------|---------|
| | | | | | | | | | | | | | | | | | | | Squa | are ro | ot (s | single | e pre | cisio | on) |
| [Instruction format] | | SQRTF | .s | reg2, | reg3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | sq | rt reg2 | 2 | | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format | F: | I | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 r 1 | rrr | 11 r | 10 1 1 | 1 1 | 1 1 | 5 . L | 4 0 (|) 0 | 0 | 0 | 31 w | w | w | 2 w v | 72 | 26 2 1 | 2 <u>5</u> 0 (| 23 | 1 1 | 21 2 0 | 20 0 1 | 1 | 17 · 1 | 16 0 |
| | | reg2 | | | | | | | | | - | | r | reg3 | | | | cate | gory | typ | e | su | b-op | | |
| [Description] | | This ins point for register rounded the resu | stru orm re 1 ir | action nat con g3. Th n accor becom | obtain tents o e oper dance es –0. | ns the of ge ration with | e ar ene n is n th | rithi ral- s ex ne c | neti purț ecut urre | c po pose ted a nt r | osit re as i oun | ive gist f it ıdir | squ ter r wei ng n | iare eg2 re o node | roo , an f inf e. W | t o d s ïni 'he | f th tor ite : n tl | e si es it accu ne so | ngle in g racy | e-preo gener y, and e op | cisio al-j d th erar | on fl purp e res nd va | oati ose sult alue | ng- is is – | 0, |
| [Floating-point operation exceptions | ;] | Unimpl Invalid Inexact | en op ex | nented eration ceptio | opera 1 exce n (I) | tion ptior | exe 1 (V | cept V) | tion | (E) | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | –Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
|------------------------------------|--------|--------------|----|----|----|--------------|-------|--------------|
| Operation result [exception] | √A | Q-NaN [V] | +0 | -0 | +∞ | Q-NaN [V] | Q-NaN | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| SUBF.D | | | | | | | | | | | | | | F | loati | ng- | -poii | nt S | ubt | ract | (Dou | ıble) |
|------------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|-------------------------------------------------|------------------------------------|---------------------------------------------|--------------------------------------------|--------------------------------------|------------------------------------|----------------------------------|-----------------------------------|-----------------------------------|------------------------------------|-------------------------------------|-------------------------------------|-----------------------------------------|--------------------------------|------------------------------------|---------------------------------------|-------------------------------|----------------------------------------|---------------------------------------|-------------------|
| | | | | | | | | | | | Flo | oati | ng- | ooin | t sub | otra | ictio | <u>n (d</u> | lou | ble p | recis | sion) |
| [Instruction format] | SUBF.D | reg1, reg2, re | g3 | | | | | | | | | | | | | | | | | | | |
| [Operation] | reg3 ← re | eg2 – reg1 | | | | | | | | | | | | | | | | | | | | |
| [Format] | Format F: | I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 15 14 | 10 | F | 4 | | | 0 | 04 | | | | 07 | 26 | 25 | | 22 | <u></u> | 04 / | 20 | | 4- | 7 46 |
| | r r r r 0 | 1 1 1 1 | 5 1 1 | 4 R | R | r r | 0 | 31 w | w | w | w | 0 | 1 | 25 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 1 | 0 |
| | reg2 | | | | re | eg1 | | | r | eg3 | | | | cat | tegoi | у | typ | е | ; | sub- | ор | |
| [Description] | This instru- pair speci format co result in th as if it we rounding | uction subtrac fied by genera ntents of the r ne register pai re of infinite a mode. | ets the al-pur registe r spec accur | e do rpo er p cifi acy | ouble se re pair s ed by r, and | e-pre egiste speci y ger d the | cisi er re fiec nera res | ion eg1 l by ll-pu ult | floa fro ge urp is r | atin m ti ner ose oun | g-p he (al-j reg dec | ooir dou pur gist d ir | nt f ible pos er i n ac | orm e-pi se r reg: ccoi | nat c recis egis 3. Th rdan | on ion ter ie o ce | tent n flo reg ope wit | ts o pati g2, a rati h th | f th ng anc on ne | ne re -poi 1 stc is e curr | egist int ores f xecu ent | er the ited |
| [Floating-point operation exceptions] | Unimplen] Invalid op Inexact ex Overflow Underflow | nented operation peration except acception (I) exception (O w exception (I | ion ex otion () J) | xce (V) | ptio | n (E) | | | | | | | | | | | | | | | | |



[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|-----|----|-------|------------|-------|-----------|
| reg1(A) | Normal | –Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
| Normal | | | | | | | | |
| -Normal | | P | ٨ | | | | | |
| +0 | | В- | - A | | +∞ | _∞ | | |
| -0 | | | | | | | | |
| +∞ | | _ | ∞ | | Q-NaN | | | |
| | | | | | [V] | | | |
| _∞ | | | +∞ | | | Q-NaN | | |
| | | | | | | [V] | | |
| Q-NaN | | | | | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.


| SUBF.S | | | | | | | | | | | | | | | F | loat | ting | g-point | Sub | otra | ct (S | ingle | ;) |
|----------------------|----|---------------------------------------------------------|------------------------------------------|-----------------------------------------|----------------------------------------------|----------------------------|--------------------------------------|-------------------------------|-------------------------------|------------------------------|----------------------|------------------------------|-----------------------------------|------------------------------|---------------------------------|-------------------------------|------------------|---------------------------------|------------------------|----------------------|--------------------------|------------------------|----------|
| | | | | | | | | | | | | | Floa | ating | I-poir | nt su | btı | raction | (sin | gle | prec | isior | 1) |
| [Instruction format] | | SUBF.S | reg1, i | reg2, r | eg3 | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← re | eg2 – 1 | reg1 | | | | | | | | | | | | | | | | | | | |
| [Format] | | Format F | I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | 11 | 10 | | 5 | 4 | | | 0 | 31 | | | 27 | 26 | 25 | 2 | 23 | 22 21 | 20 | | | 17 1 | <u>6</u> |
| | r | rrrr | | ΙI | ΙI | R | RF | R R | R | W | W | W | w w | 1 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 (| |
| | | reg2 | | | | | re | g1 | | | r | eg3 | | | cat | egor | У | type | | sub | -op | | |
| [Description] | | This instru reg1 from the result i and the res | ction s he sing n gene ult is r | ubtract gle-pre ral-pur oundec | ts the si cision f pose re l in acc | ngl loa gist core | e-prec ting-p ter reg lance | isior oint 3. Tl wit | n flo forn he o h th | oatin nat oper ne c | ng-p cont atio | oint tents n is ent | form s of ge exect round | at c ener ited ling | onter al-pu as if g mo | nts o irpos it w de. | f g se ver | general registe re of int | -pui r reg finit | pos g2, s e ac | se reg and s cours | giste store acy, | r s |
| [Floating-point | | Unimpler | nenteo | l opera | ation e | xce | eption | (E) | | | | | | | | | | | | | | | |
| operation exceptions | 5] | Invalid op | peratio | on exce | eption | (V |) | | | | | | | | | | | | | | | | |
| | | Inexact ex | cepti | on (I) | \mathbf{O} | | | | | | | | | | | | | | | | | | |
| | | Underflov | excer v exce | ntion | U) (II) | | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2(B) | | | | | | | | |
|---------|--------|---------|-----|----|-----------|-----------|-------|-----------|
| reg1(A) | Normal | –Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
| Normal | | | | | | | | |
| -Normal | | Р | ^ | | 1.00 | | | |
| +0 | | D - | - A | | +∞ | _∞ | | |
| -0 | | | | | | | | |
| +∞ | | | × | | Q-NaN [V] | | | |
| -∞ | | | +∞ | | | Q-NaN [V] | | |
| Q-NaN | | | | | | | Q-NaN | |
| S-NaN | | | | | | | | Q-NaN [V] |

Note 1. [] indicates an exception that must occur.



| TRFSR | | | | | | | Т | ransf | ers sp | ecifi | ed | CC bit | to Z | ero fla | g in P | SW (S | ingle) |
|--------------------------------------|---------------------------------------|-----------------------------------------------|----------|-------------|-------------------|----------------|---------------|--------------|------------------|----------------|------------|--------------------|--------------|-------------------|----------------|----------------|--------|
| | | | | | | | | | | | | | | | F | lag tra | Insfer |
| [Instruction format] | TRFSR 1 TRFSR | fcbit | | | | | | | | | | | | | | | |
| [Operation] | PSW.Z ← | – fcbit | | | | | | | | | | | | | | | |
| [Format] | Format F | ?:I | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | |
| | 15 11 | 10 | 5 | 4 | | | 0 3' | 1 | | 27 | 26 | 25 | 23 | 22 21 | 20 | - | 7 16 |
| | 0 0 0 0 0 | 1 1 1 1 | . 1 1 | 0 | 0 0 | 0 | 0 0 | 0 | 0 0 | 0 | 1 | 0 0 | 0 | 0 0 | 0 f | f | f 0 |
| | | | | | | | | r | eg3 | | | categ | ory | type | รเ | ıb-op | |
| | Note: fcb | oit: fff | | | | | | | | | | | | | | | |
| [Description] | This instr register s the CC0 l | ruction trans pecified by bit (bit 24). | fers the | e co the | ndition Z flag | n bit in tl | s (tł ne P | ne CO SW. | C(7:0) If fcb |) bit it is | s: t on | oits 31 nitted, | to , this | 24) in s instr | the I uctio | FPSR n tran | sfers |
| [Floating-point operation exceptions | None | | | | | | | | | | | | | | | | |



| TRNCF.D | L | | | | | | | | F | loa | iting | I-bo | int | Cor | ז∨∈ | ert D | oub | le to |) Lo | ng, rc | un | d to | ward | d z | ero (| Dou | ble) |
|-----------------------------------------|----|----------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------|------------------------------------------------------------------------------|-------------------------------------------------------------------------------|-----------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|------------------------------------------------------------------------|-------------------------------------------------------------------------|--------------------------------------------------------|-----------------------------------------------------------------|--------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------|----------------------------------------------------------------------------------------------|----------------------------------------------|-------------------------------------------------------------------------|-------------------------------------------|---------------------------------------------|---------------------------------------------------|------------------------|-----------|
| | | | | | | | | | | | | | | С | on | vers | ion | to fi | xed | -point | fo | rma | t (dc | bub | le pi | recis | ion) |
| [Instruction format] | | TR | NCF.I | DL re | g2, r | eg3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg | 3 ← t | runc r | eg2 (| doul | ble - | →]e | ong | -W | ord |) | | | | | | | | | | | | | | | |
| [Format] | | For | mat F | :I | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | _ | _ | | | | | | | | | | | | | | | | _ | | | |
| | 15 | | 11 | 10 | - 1 | 1 1 | 5 | 4 | 0 | 0 | 0 | 0 | 31 | | | | 27 | 26 | 25 | 2 | 32 | <u>22 2</u> | 1 20 | 0 | 0 1 | | 16 |
| | r | r r | r U | | Ţ | 1 1 | | 0 | 0 | 0 | 0 | T | W | W | W | W | 0 | | 0 | 0 (|) | 1 (| | - | 0 1 | . 0 | 0 |
| | | reg2 | 2 | | | | | | | | | | | I | reg | j 3 | | | ca | tegor | / | type | ; | s | ub-o | р | |
| [Description] | | Thi the stor The Wh ran If in reg foll | s instr regist res the e result en the ge of 2 nvalid ister is lows, a Sour | ruction er pair e resul t is ro e source $2^{63} - 2^{63}$ opera s set a accord rce is a rce is a | n arit r spe t in t unde ce op 1 to - ation s an ling t a pos | hme cifie he red in eran -2^{63} , exce inva to di sitive gativ | tical d by egist the d is , an lid c ffere e num e num | ly c y ge zer p zerc infi IEE ons = oper ence mbe | conv ner o di nite EE7: are atic es a: er o: er, 1 | vert al- I spe rec or 54- not mo r + not | ts the purple of the purple o | ne c pos fiec n, r t-a- ine abl no sou 2 ⁶³ nur | lou e re l by ega -nu d in led, ex urce - 1 | ble egi: 7 ge ardl mb nva , the cep es. 1 is | e-p ste ene les oer alic e p otic | reci er re eral- s of , or 1 op prese on o | sion g2 pure f the era erva erva erva a | n fl to 6 rpos e cu en t tion atio urs. | oati 54-t se r he i n ex on b The s re | ng-p bit fix egist nt roo cound ccepti it (bi it (bi it (bi turne | oin er und lec ior t 4 urn | nt fo l-po reg: ding l res n is n is n is n val | orma int 3. g mo sult dete | at for odd is ect e F dif | cont rma e. outs ed. 'PSF ffers | tents t, an side | s of d |
| [Floating-point operation exceptions |] | Uni Inv Ine | imple alid o xact e | mente peratio xcepti | d ope on ex ion (1 | eratio ccept | on e tion | xce (V) | ptic | on (| (E) | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | _ ∞ | Q-NaN | S-NaN |
|------------------------------------|--------|---------|--------|-------|----------------|------------|--------------|-------|
| Operation result [exception] | A (int | eger) | 0 (int | eger) | Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| TRNCF.D | UL | | | F | loat | ing-p | point | Cor | Conv | t Do | sion | to une | nsig | inec ed fi | ixed | ng, rou -point | und forn | tow | ard (dou | zero ıble <u>ı</u> | precis | ible) sion) |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|---------------------------------------------------------------------------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|----------------------------------------------------------------------------|-----------------------------------------------------------------|----------------------------------------------------------------------------------------|-------------------------------------------------------------------|-----------------------------------------------------------|--------------------------------------------------------------|-----------------------------------------------------------------|------------------------------------------------------|----------------------------|
| [Instruction format] | TRNCF. | DUL 1 | reg2, r | eg3 | | | | | | | | | | | | | | | | | | |
| [Operation] | reg3 ← | trunc re | eg2 (do | ouble | e → | • un | sign | ed 1 | ong | g-w | orc | l) | | | | | | | | | | |
| [Format] | Format l | F:I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | _ | | | | | ~ | | | | | 0.5 | | | | | | 4- | - 40 |
| | 15 1 15 1 | 1 10 | 1 1 | 1 | 1 | 4 | <u>)</u> | 0 | 0 | 31 | 747 | X47 X47 | 27 | 26 | 25 | 23 | 1 22 | <u>21</u> | 20 | 0 | 17 | / 16 |
| | | | | 1 | - | I (| 5 0 | 0 | 1 | vv | vv | w w | 0 | | 0 | 0 0 | 1 | | | | 1 0 | |
| [Description] | This inst the regis format, a The resu When th result is detected If invalia register follows, • Sou • Sou | ruction ter pain and sto- ilt is ro- e source outside d opera is set a: accord urce is a urce is a | n arithr r speci res the unded ce oper e the ra ation ex s an in ling to a posit a negat | netic fied 1 resu in th rand unge kcept valid diffe ive n | eally by ilt i ie z is i of 2 tior tior l op erer | y con gene n the ero o nfin 2 ⁶⁴ - ns ar perat nces uber nber | nver eral- e reg direc ite, 1 - 1 t e no tion a amo outs ; not | ts the purgisted state of the purgisted state | he c pos er p n, r a-n , an abl no sou the | lou e r pair rega um IE led, ex urce ra nbe | ble egis spardl ber EE , the cep es. nge | -prec: ster re- ecified ess or r, or n 754-c e pres- ption c e of 2° or $-\infty$: | ision eg2 d by f the ega lefir erv: $5^{54} -$ 0 is | n fl to u 7 ge e cu tive ned atic urs. 1 t s re | oati insi ener irre e nu inv on b Th Th | ing-po igned ral-pur umber valid c vit (bit e retu , or + ned. | oint 64- rpos ndi , or oper 4) rn v xo: 2 | for bit se r mg : wh catic of t valu | mat fixe egis moo len t on e the 1 le d | t cor ed-p ster de. the r exce iffer is re | ntent oint reg3 roun ptior R rs as | s of ded n is ed. |
| [Floating-point operation exceptions] | Unimple] Invalid o Inexact o | ementeo operatio excepti | d opera on exce on (I) | ation eptio | ex on (| cept V) | tion | (E) | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|-----------|-------|
| Operation result [exception] | A (Integer) | 0 [V] | 0 (Int | eger) | Max U-Int [V] | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| TRNCF.D | U | W | | | | | FI | oa | ting- | poii | nt C | Conv | 'er | 't Dc | bul | ole to |) Ur | sig | nec | I-W | /ord, r | ound | d to | owa | rd z | ero | (Doi | uble) |
|--------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|-----------------------------------------------------------------|-------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|--------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|--------------------------------------------------------------------------------------|------------------------------------------------|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|-----------------------------------------------------------------|---------------------------------------------------|---------------------------------------------------------------------------|-----------------------------------------------------|---------------------------------------------|--------------------------------------------------|--------------------------------------------------------|-------------------------------------------------------|-------------------------------|----------------|
| | | | | | | | | | | | | Сс | on | vers | io | n to | unsi | gn | ed f | ixe | d-poin | t for | rma | at (d | lout | ole p | reci | sion |
| [Instruction format] | | TRNC | F.C | OUW | reg | 2, re | g3 | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 ← | – tr | unc r | eg2 | (dou | ıble - | \rightarrow | unsi | ign | ed | woi | rd |) | | | | | | | | | | | | | | |
| [Format] | | Forma | t F: | I | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | 40 | | | _ | | | | | 0 | 0 | | | | 0 | | | | 00 | 00 | | | | | 47 | 10 |
| | 15 r | rrr | 11 0 | 10 | 1 | 1 | 5 1 1 | 4 | | 0 | 0 | 1 | 3 | 81 w/ w/ | 7 | w v | 2 | 72 | 1 1 | <u>.5</u> | 0 0 | 1 | 21 | 1 20 | 0 | 0 | 17 0 | 16 |
| | Ĺ | | 0 | | | - | | | | 0 | | - | ' | | | | | | - | | | | | | | | | |
| [Description] | I | This in the reg format The re When result i detecte If inva registe follow • So • So | istru ista , ar sulf the is o ed. lid ar is s, a our | uction er pai ad sto t is ro sour utsid opera set a ccoro ce is ce is | n ari ir spe ores t bound ce op e the ation as an ding a po a ne | thme ecific the ro ed in peran e ran inva to di sitiv gativ | etical ed by esult i the nd is ge of alid c ifferent e nu ve nu | lly g in ze in f 2 ons ope enc mb | con gener i gen ro d finit 3^2 _ s are eration ces a per c ber, | ver ral- irea irea 1 t a no on umo outs no | rts = -pu al-p ctic not to (ong side t-a- | the orpose ourpon, 1 on, | dc se reg nu n I le ur e 1 ml | oubl reg se r garc mbo EEI d, tl excee cess. rang ber, | le- tis eg til er E7 he p | pred ter 1 tiste ess (, or $^{-2}$ pre tion of 2 r $-\infty$ | cision $reg2$ r record the reg2 of the record the reco | on g3 ati ine vati cur - 1 is | floa un curr ve 1 d ii ion s. T to retu | ati isi rer nu bi The 0, urn | ng-po gned i mber, alid o it (bit e retur or +0 ned. | int : 32-1 ndin or pera 4) (:n v | for bit ng wh ati of falu | rma fix mo hen on c the ue d | t co ed- ode. the exc FP liffo is | onte poin e rou epti 'SR ers a retu | nts nt unde on is | of ed is |
| [Floating-point operation exceptions |] | Unimp Invalic Inexac | olen l op t ex | nente berati kcept | ed op on e ion (| oerati xcep []) | ion e otion | xc (V | eptio 7) | on | (E) |) | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (Integer) | 0 [V] | 0 (Int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.



| TRNCF.D | W | ļ | | | | | | | | | Floa | ating | -poi | int (| Con | vert | Doul | ole t | o W | ord, r | ou | nd to | war | d z | ero (| Dout | ole) |
|--------------------------------------|---------|----------------------------------------------------------|-----------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------|----------------------------------------------------------------------------|------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|---------------------------------------------------------------|--------------------------------------------------------------------------------|-----------------------------------------------------------------|--------------------------------------------------------|---------------------------------------------------------------------|----------------------------------------------|---------------------------------------------------------------|-------------------------------------------|---------------------------------------------------------------|---------------------------------------------------|------------------------------------------|---------------------------------------------------|------------------------------|---------|
| [Instruction format] | | TF | RNC | F.D | OW r | eg2 | , reg | g3 | | | | | | | <u> </u> | onve | rsior | n to | lixeo | <u>1-poin</u> | it fo | ormat | <u>t (d</u> | <u>oub</u> | <u>le pr</u> | ecisi | on) |
| [Operation] | | reg | g3 ← | – tr | unc r | eg2 | (do | uble | \rightarrow | wo | ord) | | | | | | | | | | | | | | | | |
| [Format] | | Fo | orma | t F: | I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | _ | _ | | | | | | | | | | | | | | | | | |
| | 15 r | r r | r | <u>11</u> 0 | 10 1 1 | . 1 | 1 | 1 : | 5 4 1 (| 1 | 0 0 | 0 | 0 | 31 w | w | w | 2 w v | 720 | 0 | 0 | 2 <u>3</u> 0 | 22 2 1 (| 12 0 | 0 1 | 0 0 | <u>17</u> | 16 0 |
| | | reç | J2 | | | | | | | | | | | | 1 | reg3 | | | Са | atego | у | type | 3 | s | ub-o | p | |
| [Description] | | Th the sto Th W ran If : reg fol | tis in e reg ores t he res hen t nge c inva giste llow S S S | astruction giste the sulf the of 2 lid er is s, a our our | uction er pai resul t is rc sour- y^{31} – opera set a ccord ce is ce is | n ari r sp lt in ound ce o 1 to ation s an ding a pc a ne | ithm ecif gen led i pera -2^3 n exa i inv ; to c ositi egati | netica ied b eral- in the and is ³¹ , ar cepti valid differ ve nu | ally y g pure ze s in IE ions ope rend uml | co gene rpo ro fini EE s ar erat ces ber ber | eral- se re directile of 2754 re no tion amo or + | ets the purposed of the purpo | he c pos ter : n, r t-a- ïne abl no sou 2 ³¹ num | lou e r reg ega -nu d i led, urce - 1 | ble egis 3. ardl mb nva , tho cep es. 1 is er, c | -pre ster ess er, c lid c e pro otion retu | cisid reg2 of th r wh oper eserv occ | on f to ne c nen atic vati turs 1. 2^{31} | loat 32-1-1 the the on t . Th | ing-j bit fi ent rc roun xcep bit (b be ret | ooi xe our de tio it ur | int fc d-po nding cd res n is (4) of n val | orm pint g m sult det f the lue | at of for odd is of ecture F | cont rmat e. outs ed. PSR ffers | ents , and ide t as | of d |
| [Floating-point operation exceptions |] | Ur Inv Ine | imp valic exac | olen 1 op 2t ex | nente berati kcept | d op on e ion | pera exce (I) | tion ptioı | exc 1 (V | ept 7) | tion | (E) | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -8 | Q-NaN | S-NaN |
|------------------------------------|--------|---------|--------|-------|----------------|----|--------------|-------|
| Operation result [exception] | A (Int | eger) | 0 (Int | eger) | Max Int [V] | | –Max Int [V] | |

Note 1. [] indicates an exception that must occur.



| TRNCF.S | L | | | | | | | | | Flo | atin | ıg-p | ioc | nt C | Cor | nvert | Sin | gle | to L | ong, re | ound | tow | rard | zero |) (Sir | ngle) |
|--------------------------------------|----|---------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|------------------------------------------------------------------|----------------------------------------------------------------------------------|-------------------------------------------------------------------|-------------------------------------------------------------------------|-----------------------------------------------------------|------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------|------------------------------------------------------|-------------------------------------------------------|---------------------------------------------------------------------|-------------------------------------------------------------------|---------------------------------------------------|-----------------------------------------------------|------------------------------------------|-----------------------------------|---------------------|
| | | | | | | | | | | | | | | | Сс | onve | sior | n to | fixe | d-poin | t forr | nat | (sin | gle p | recis | sion) |
| [Instruction format] | | TRN | CF.S | SL reg | 2, reg | g3 | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | reg3 | ← tı | unc re | eg2 (s | ingle | ; — | → lo | ong-v | voi | rd) | | | | | | | | | | | | | | | |
| [Format] | | Form | at F | :I | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | 45 | | | 4.0 | | | _ | | | | | • | | | | | 07 | | 05 | | | 04 | 00 | | 4- | 7 40 |
| | 15 | | 11 | 10 | 1 1 | 1 1 | 1 | 4 | 0 | 0 | 0 | 1 | 31 | | | | 27 | 26 | 25 | 2 | 3 22 | 21 | 20 | 0 | 11 | / 16 |
| | r | r r i reg2 | r r | | 1 1 | | 1 | U | 0 | 0 | U | 1 | w | w | re | w w 93 | 0 | | ca | itegory | ty | pe | 0 | sub-(| op | 0 |
| [Description] | | This gener pair s The r When range If inv regist follow | instr ral-p ppeci esul n the e of 2 ralid ter is ws, a Sour Sour | uction urpose fied b t is rou source $2^{63} - 1$ opera s set as accord rce is a rce is a | arith e regi y gen undec e ope to – tion e an in ing to posi a nega | ster r eral- l in th rand 2 ⁶³ , a excep nvalid diffu tive r ative | cal pun is i un l utio d o ere nur nu | ly o 2 to rpo zero infi IEE ns per ence mbe | conv o 64- se re o dir nite EE75 are r ration es an er or er, n | ert bit gis ect or 4-c not not +o ot- | s th fix ster tior not leff ena nd s s c: 2 a-n | ne ked re n, r z-a- ine abl no sou 263 | sin l-p- g3 reg -nu ed i led ex urc - nbe | igle oin ard inta , th acej es. 1 is | e-p at f lle bei ali ne pti s r | For the second | sion at, f the who oera erv occu ned -2 | n flo and e cu en t tion atic urs. | oati stc he he n ex on b Th s re | ing-po pres th nt rou round ccepti bit (bi e retu | bint : ne re undin ed r on is : 4) o rn v d. | for sult ng 1 esu s de of t alu | nat t in moo lt is etec he l e di | con the de. out ted. FPSI | tent regi side R s as | s of ster the |
| [Floating-point operation exceptions |] | Unim Inval Inexa | npler id oj ict ez | nenteo peratic xcepti | l oper on exc on (I) | ratior | n ez on | xce (V) | ptioi) | n (1 | E) | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | _∞ | Q-NaN | S-NaN |
|------------------------------------|--------|---------|--------|-------|----------------|----|--------------|-------|
| Operation result [exception] | A (Int | eger) | 0 (Int | eger) | Max Int [V] | | -Max Int [V] | |

Note 1.] indicates an exception that must occur.



| TRNCF.S | U | L | , | | | | | | | Fl | oati | ing- | poi | nt C | Con | /ei | rt S | Sing | gle | to l | Jnsi | gn | əd-L | _or | ng, ro | unc | l to | ware | d ze | ro (| Sing | gle) |
|------------------------------------------|----|-------------------------------------------------|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|------------------------------------------------------------------------------|-----------------------------------------------------------------------|------------------------------------------------------------------------------|------------------------------------------------------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|-----------------------------------------------------------------------|---------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|--------------------------------------------------------------------|--------------------------------------------------------------------------|-----------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------|---------------------------------------------------|--------------------------------------------|---------------------------------------------------------------------------------------|--------------------------------------------------------|--------------------------------------|--------------------------------------|---------------------------------------------|---------------------------------------------|--------------------------------|----------------|
| | | | | | | | | | | | | | | | Co | nv | ers | sior | n to | o un | sigr | ed | fixe | ed- | -point | for | ma | t (si | ngle | pre | cisi | on) |
| [Instruction format] | | Г | ΓRN | ICF.S | SUL | re | eg2 | , re | g3 | | | | | | | | | | | | | | | | | | | | | | | |
| [Operation] | | r | eg3 | ← tı | unc | re | g2 | (sin | ıgle | e — | → u | nsig | gne | ed l | ong | ç-v | WO: | rd) |) | | | | | | | | | | | | | |
| [Format] | | F | Form | nat F | :I | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 | | | 11 | 10 | 1 | - | 1 | 1 | 5 | 4 | | 0 | 0 | 0 | 3 | 31 | | | | 27 | 26 | 3 25 | 5 | 23 | 22 | <u>2</u> | 1 20 | 0 | | 17 | 16 |
| | r | r | r | r r | | Ţ | Ţ | T | T | Ţ | | 0 | 0 | 0 | Ţ | 1 | W | W | W | W | 0 | | 0 | | 0 0 | | 0 | 0 | 0 | 1 | 0 | 0 |
| | ļ | r | eg2 | | | | | | | | | | | | | l | | I | reg | 13 | | | Ca | ate | egory | ty | γpe | | sul | э-ор | | |
| [Description] | | T g tl T V r d I r f | This gene he r The Whe result leteo f in regis Follo | instr eral-p egist resul en the lt is c cted. valid ster is ows, a Sour Sour | ucti urpe er p t is : sou outsi ope s set acco rce i | on ose air rou irce ide erat : as ordi s a s a | ari reg specinde e op the ion an ng po neg | thm gist ecif ed i pera ran e ran to c sitiv gati | neti er 1 ied n tl and nge cep vali liff ve | cal reg2 by he 2 is of otio d o čere nur nu | ly 2 to 2 to 2 cor inf 2 2 ⁶ ons open cons mbo | con o ur ener o d init 4 _ are ratio es a er o per, | ven nsig al- irec e, 1 1 t no on uts no | rts gne -pu: ctic not to 0 of e and ong side t-a- | the ed 6 rpo: on, 1 -a-1), ar nab 1 nc ; so 1 nc ; so -nu | si 4- se re nu n I ble o e ur e 1 ml | ing bit re gan ml IEF ed, exc rces ran bei | t fi gis rdl ber EE the s. nge | -pr ixe ste les r, c 75 e p otic e o | reciped-per reciper r | sion opin eg3. f the ega lefin erv occu | n fl t f e c tiv nec ativ urs 1 s r | oat orm e v l in on l . Th to 0 | tin nat en alu va bit ne | g-po t, and t rou ue, o ulid o t (bit return or $+\infty$ ed. | int l st ndi r w per 4) rn v o: 2 | for ore he ati of val | rma es the moon of the ue of $r = 1$ | t co be ro e ro exc FP liffe | onte esul ound epti SR ers a | nts t in ded on as | of is d. |
| [Floating-point operation exceptions] |] | U I I | Jnir nva nex | nplei lid oj act e | nen pera xcej | ted tio | op ne: on (| era xce I) | tio ptio | n ex | xce (V) | eptio) | on | (E) |) | | | | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (Integer) | 0 [V] | 0 (Int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.

| TRNCF.S | WU | | | Floa | atin | ng-p | oint (| Conv | ert | Sin | gle | to U | Insię | gne | d-W | ord, ro | und | towa | ard | zero | (Sin | gle) |
|--------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------|-----------------------------------------------------|-------------------------------------------------------|-----------------------------------------------------|-------------------------------------------------------|---------------------|----------------------------------------------|-------------------------------------------|-------------------------------------------------------------------|-----------------------------------------------------|------------------------------------------------------|-------------------------------------|--------------------------------------------------|-----------------------------------|------------------|
| | | | | | | | | Со | nve | ersic | on to | o un | sign | ed | fixe | d-point | forn | nat (| sin | gle p | recis | ion) |
| [Instruction format] | TRNCF.S | SUW r | eg2, reg | 3 | | | | | | | | | | | | | | | | | | |
| [Operation] | reg3 ← tı | unc reg | g2 (sing | $le \rightarrow$ | un | sig | ned | vore | 1) | | | | | | | | | | | | | |
| [Format] | Format F | :I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | | | | | | | | |
| [0] | 15 11 | 10 | | 5 | 4 | | | 0 | 3. | 1 | | | 27 | 26 | 25 | 23 | 22 | 21 2 | 20 | | 17 | 16 |
| | rrrrr | 1 1 | 1 1 1 | . 1 | 1 | 0 | 0 0 | 1 | W | w | W | w | W | 1 | 0 | 0 0 | 1 | 0 | 0 | 0 0 | 0 | 0 |
| | reg2 | | | | | | | | | | reg | 3 | | | cat | tegory | typ | be | : | sub-o | р | |
| | | Ι | | ļ | | | | | ļ | | 0 | | | | I | 0, | | I | | | • | |
| [Description] | This instr contents of result in g The resul When the result is of detected. If invalid register is follows, a | uction of gene general- t is rou source outside operation set as according | arithme ral-purp -purpose nded in e operan the rang ion exce an inval ng to dif | tically ose regi the zeri d is in ge of 2 eption lid op | y c egi ste ero nfin 2 ³² s a era ces | onv ster r re din nite $\frac{1}{2} - 1$ are 1 atio | verts r reg g3. rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rection rectio | the 2 to 2 to 2 to 2 to 2 to 2 to 2 to 2 to | sin un reg nun 11 lec o ez urc | ngle nsign gard EEE l, th xcep xes. | e-pr ned less er, c 275 ne p ptic | ecis 132 s of or ne 4-de orese on o | the egat erva | flo fix cu ive ed utio rs. | nrren nrren nu inv n b The | ng-po point nt rou mber, alid o it (bit e retur | int r forr ndir or pera 4) c m va | num nat, ng m whe atior of th alue | be an noc n t n e di | r for ad sto le. the ro xcep FPSR | mat ores bund tion as | the led is |
| | • Sour | ce is a | positive | e num | be | r oı | ıtsid | e the | e r | ang | e o | f 2 ³ | 2_ | 1 to | o 0, | or +0 | o: 2 ³ | 32 _ | 1 | is ret | urne | ed. |
| | • Sour | ce is a | negative | e nun | ıbe | er, r | not-a | -nur | nb | er, | or - | -∞: | 0 is | s re | turi | ned. | | | | | | |
| [Floating-point operation exceptions | Unimpler] Invalid oj Inexact ez | nented peration cceptio | operation n except n (I) | on ex tion (| cep V) | otio | n (E |) | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|----------------|---------|--------|-------|------------------|----|-------|-------|
| Operation result [exception] | A (Integer) | 0 [V] | 0 (Int | eger) | Max U-Int [V] | | 0 [V] | |

Note 1. [] indicates an exception that must occur.



| TRNCF.S | W | | | | | | | | FI | oatir | ıg-p | oint | t Co | onvert | Sing | gle t | o W | ord, ro | forr | tow | /ard /sir | zero | (Sinę | gle) ion) |
|--------------------------------------|-----------|------------------------------------------------------------------------------------|-------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------|------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------|--------------------------------------------------------------------------|------------------------------------------------------------------------|---------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------|------------------------------------------------------|---------------------------------------------------------------|-------------------------------------------------------------------------------|--------------------------------------------------------|----------------------------------------------------|------------------------------------------------|-----------------------------------------------------|-----------------------------|--------------|
| [Instruction format] | | TRN | CF.S | SW re | eg2, r | eg3 | | | | | | | | | 5101 | 110 | | | 1011 | nat | (011) | gie pi | 00101 | |
| [Operation] | | reg3 • | — tı | runc r | eg2 (s | single | $e \rightarrow$ | · wo | ord) | | | | | | | | | | | | | | | |
| [Format] | | Form | at F | :I | | | | | | | | | | | | | | | | | | | | |
| [Opcode] | | | | | | | | | | | | | | | _ | | | | | | | | | |
| | 15 r 1 | rrr | 11 r | 10 | 1 | 1 1 | 5 1 | 4 0 | 0 0 | 0 | 1 | 31 w | w | w w | 27 w | 1 | 25 0 | 0 0 | 1 | 21 0 | 20 0 | 0 0 | <u> </u> | 16 0 |
| | | reg2 | | | | | | | | | | | I | eg3 | | | cat | tegory | ty | ре | | sub-o | p | |
| [Description] | | This is conte gener The re When range If inv regist follow | nstr nts (al-p esul of 2 alid er is vs, a Sour | uction of gen urpos t is ro source 2^{31} – opera s set a accord rce is rce is | a arith eral-j e regi under e ope l to – tion e s an i ling te a posi a neg | nmeti purpo ister 1 d in tl erand 2^{31} , ϵ excep nvali o diff itive 1 ative | call ose 1 reg3 is i is i an I otion d op feren nun nun | ly c reg 3. zerc nfii EE ns a pera nce nbe | conve ister i o dire nite o E754 are no ation es amo er or + er, no | rts t reg2 ctio r no -def ot en and ong -∞: : t-a-1 | he : to n, r t-a- ine abl no sou 2 ³¹ | sing 32- ega nun d in led, exc urce - 1 nbe | gle- bit nrdl nva the cep es. is r, c | precision of the precision of the precision of the present of the present of the precision | sion l-pc f the who pera erv pccu ned z = -2 | n fle int e cu en t tion atic urs. | oati: form nrren he r n ex on b The S re | ng-po mat, a nt rou coundo ceptio it (bit e retur turneo | int n nd : ndin ed r on is 4) (rn v | num stor ng 1 esu s de of ti alu | nbe res mod lt is etec he l e d | er for: the re de. 3 outs cted. FPSF | mat esult ide t as | : in the |
| [Floating-point operation exceptions | ;] | Unim Invali Inexa | pler d oj ct e: | nente peratio xcepti | d ope on ex on (I) | ration ception | n ex on (| (V) | ption | (E) | | | | | | | | | | | | | | |

[Operation result]

| reg2 (A) | Normal | -Normal | +0 | -0 | +∞ | -∞ | Q-NaN | S-NaN |
|------------------------------------|--------|---------|--------|-------|----------------|----|--------------|-------|
| Operation result [exception] | A (Int | teger) | 0 (Int | eger) | Max Int [V] | | -Max Int [V] | |

Note 1. [] indicates an exception that must occur.



Section 8 RESET

8.1 Status of Registers After Reset

If a reset signal is input by a method defined by the hardware specifications, the program registers and system registers are placed in the status shown by the value after reset of each register in **Section 3**, **REGISTER SET**, and program execution is started. Set the contents of each register to an appropriate value in the program.

The CPU executes a reset to start execution of a program from the reset address specified by **4.4**, **Exception Handler Address**.

Note that because the PSW.ID bit is set (1) immediately after a reset, conditional EI level exceptions will not be acknowledged. To acknowledge conditional EI level exceptions, clear (0) the PSW.ID bit.



APPENDIX A HAZARD RESOLUTION PROCEDURE FOR SYSTEM REGISTERS

Certain system registers require the following procedures to resolve hazards when their values are updated in the case of the instructions listed below.

• Instruction fetching

When an instruction is to be fetched after updating a register covered by the description below, after executing the instruction to update the register, only allow the instruction fetch to start after execution of an EIRET, FERET, or SYNCI instruction.

- PSW.UM, MCFG0.SPID

When an instruction is to be fetched after updating a register covered by the description below, execute the instruction to update the register before allowing the instruction fetch to start.

- All registers related to ASID and MPU (register number : SR*, 5-7)
- Load/Store

When an instruction associated with Load/Store after updating the registers below, execute a SYNCP instruction after executing the instruction to update the registers before Load/Store instruction.

- ASID, MPU protection area setting register (Register number: SR*,6–7)
- Interrupt

Update the registers below when interrupt is inhibited. (PSW.ID = 1).

- PSW.EBV, EBASE, INTBP, ISPR, PMR, ICSR, INTCFG
- Coprocessor instruction

When a coprocessor instruction (floating-point operation instruction) is to be executed after updating the register below, execute instructions of EIRET, FERET or SYNCI after executing the instruction to update the registers and before executing a coprocessor instruction.

- PSW.CU0



APPENDIX B NUMBER OF INSTRUCTION EXECTUION CLOCKS

B.1 Numbers of Clock Cycles for Execution

Numbers of clock cycles for execution are given in this section. Since the G3MH has a pipe-lined architecture that differs from that of other CPUs, the various values given cannot be treated in a uniform manner. Moreover, the number of clock cycles required to execute an actual instruction may differ with the state of execution of the previous and next instruction.



B.2 Number of G3MH Instruction Execution Clocks

| | | | | Instruction | Number | of Executio | on Clocks |
|-----|----------------------|-----------|------------------------|-------------------|--------|-------------|-----------------|
| | | | | Length (Number | | | |
| Ту | pes of Instructions | Mnemonics | Operand | of Bytes) | issue | repeat | latency |
| Lo | ad instruction | LD.B | disp16 [reg1] , reg2 | 4 | 1 | 1 | 3 ^{*1} |
| | | | disp23 [reg1], reg3 | 6 | 1 | 1 | 3 ^{*1} |
| | | LD.BU | disp16 [reg1] , reg2 | 4 | 1 | 1 | 3 ^{*1} |
| | | | disp23 [reg1] , reg3 | 6 | 1 | 1 | 3 ^{*1} |
| | | LD.H | disp16 [reg1] , reg2 | 4 | 1 | 1 | 3 ^{*1} |
| | | | disp23 [reg1] , reg3 | 6 | 1 | 1 | 3 ^{*1} |
| | | LD.HU | disp16 [reg1] , reg2 | 4 | 1 | 1 | 3 ^{*1} |
| | | | disp23 [reg1] , reg3 | 6 | 1 | 1 | 3 ^{*1} |
| | | LD.W | disp16 [reg1] , reg2 | 4 | 1 | 1 | 3 ^{*1} |
| | | | disp23 [reg1] , reg3 | 6 | 1 | 1 | 3 ^{*1} |
| | | LD.DW | disp23 [reg1] , reg3 | 6 | 1 | 1 | 3 ^{*1} |
| | ep relative | SLD.B | disp7 [ep] , reg2 | 2 | 1 | 1 | 3 ^{*1} |
| | | SLD.BU | disp4 [ep] , reg2 | 2 | 1 | 1 | 3 ^{*1} |
| | | SLD.H | disp8 [ep] , reg2 | 2 | 1 | 1 | 3 ^{*1} |
| | | SLD.HU | disp5 [ep] , reg2 | 2 | 1 | 1 | 3 ^{*1} |
| | | SLD.W | disp8 [ep] , reg2 | 2 | 1 | 1 | 3 ^{*1} |
| Sto | I pre instrucrion | ST.B | reg2, disp16 [reg1] | 4 | 1 | 1 | 1 |
| | | | reg3, disp23 [reg1] | 6 | 1 | 1 | 1 |
| | | ST.H | reg2, disp16 [reg1] | 4 | 1 | 1 | 1 |
| | | | reg3, disp23 [reg1] | 6 | 1 | 1 | 1 |
| | | ST.W | reg2, disp16 [reg1] | 4 | 1 | 1 | 1 |
| | | | reg3, disp23 [reg1] | 6 | 1 | 1 | 1 |
| | | ST.DW | reg3, disp23 [reg1] | 6 | 1 | 1 | 1 |
| | ep relative | SST.B | reg2, disp7 [ep] | 2 | 1 | 1 | 1 |
| | | SST.H | reg2, disp8 [ep] | 2 | 1 | 1 | 1 |
| | | SST.W | reg2, disp8 [ep] | 2 | 1 | 1 | 1 |
| Mu | ltiplication | MUL | reg1, reg2, reg3 | 4 | 1 | 1 | 3 |
| ins | struction | | imm9, reg2, reg3 | 4 | 1 | 1 | 3 |
| | | MULH | reg1, reg2 | 2 | 1 | 1 | 3 |
| | | | imm5, reg2 | 2 | 1 | 1 | 3 |
| | | MULHI | imm16, reg1, reg2 | 4 | 1 | 1 | 3 |
| | | MULU | reg1, reg2, reg3 | 4 | 1 | 1 | 3 |
| | | | imm9, reg2, reg3 | 4 | 1 | 1 | 3 |
| | Multiply-accumulate | MAC | reg1, reg2, reg3, reg4 | 4 | 2 | 2 | 4 |
| | operation | MACU | reg1, reg2, reg3, reg4 | 4 | 2 | 2 | 4 |

(1) Basic instruction

RH850G3MH Software

APPENDIX B NUMBER OF INSTRUCTION EXECTUION CLOCKS

| | | | | Instruction | Number | of Executio | n Clocks |
|----|----------------------|-----------|------------------------|-------------------|--------|-------------|----------|
| | | | | Length (Number | | | |
| Ту | pes of Instructions | Mnemonics | Operand | of Bytes) | issue | repeat | latency |
| Ar | ithmetic instruction | ADD | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | ADDI | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |
| | | CMP | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | MOV | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | | imm32, reg1 | 6 | 1 | 1 | 1 |
| | | MOVEA | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |
| | | MOVHI | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |
| | | SUB | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | SUBR | reg1, reg2 | 2 | 1 | 1 | 1 |
| | Operation with | ADF | cccc, reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | condition | SBF | cccc, reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | Saturated operation | SATADD | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | | reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SATSUB | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | | reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SATSUBI | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |
| | | SATSUBR | reg1, reg2 | 2 | 1 | 1 | 1 |
| Lo | gical instruction | AND | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | ANDI | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |
| | | NOT | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | OR | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | ORI | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |
| | | TST | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | XOR | reg1, reg2 | 2 | 1 | 1 | 1 |
| | | XORI | imm16, reg1, reg2 | 4 | 1 | 1 | 1 |



RH850G3MH Software

APPENDIX B NUMBER OF INSTRUCTION EXECTUION CLOCKS

| | | | | Instruction | Number | of Executio | n Clocks |
|------|--------------------|-----------|------------------------|--------------------------------|--------|-------------------|-------------------|
| Тур | es of Instructions | Mnemonics | Operand | Length (Number of Bytes) | issue | repeat | latency |
| Dat | a operation | BINS | reg1, pos, width, reg2 | 4 | 1 | 1 | 1 |
| inst | ruction | BSH | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | BSW | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | CMOV | cccc, reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | | cccc, imm5, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | HSH | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | HSW | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | ROTL | imm5. reg2. reg3 | 4 | 1 | 1 | 1 |
| | | | reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SAR | reg1, reg2 | 4 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | | reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SASF | cccc. reg2 | 4 | 1 | 1 | 1 |
| | | SETF | cccc. reg2 | 4 | 1 | 1 | 1 |
| | | SHL | reg1, reg2 | 4 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | | reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SHR | reg1, reg2 | 4 | 1 | 1 | 1 |
| | | | imm5, reg2 | 2 | 1 | 1 | 1 |
| | | | reg1, reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SXB | reg1 | 2 | 1 | 1 | 1 |
| | | SXH | reg1 | 2 | 1 | 1 | 1 |
| | | ZXB | reg1 | 2 | 1 | 1 | 1 |
| | | ZXH | reg1 | 2 | 1 | 1 | 1 |
| Bit | search instruction | SCH0L | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SCH0R | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SCH1L | reg2, reg3 | 4 | 1 | 1 | 1 |
| | | SCH1R | reg2, reg3 | 4 | 1 | 1 | 1 |
| Divi | ision instruction | DIV | reg1, reg2, reg3 | 4 | 1 | 19 | 19 |
| | | DIVH | reg1, reg2 | 2 | 1 | 19 | 19 |
| | | | reg1, reg2, reg3 | 4 | 1 | 19 | 19 |
| | | DIVHU | reg1, reg2, reg3 | 4 | 1 | 19 | 19 |
| | | DIVU | reg1, reg2, reg3 | 4 | 1 | 19 | 19 |
| | High-speed divide | DIVQ | reg1, reg2, reg3 | 4 | 1 | N+3 ^{*2} | N+3 ^{*2} |
| | operation | DIVQU | reg1, reg2, reg3 | 4 | 1 | N+3 ^{*2} | N+3 ^{*2} |



APPENDIX B NUMBER OF INSTRUCTION EXECTUION CLOCKS

| | | | Instruction | Number | of Executio | n Clocks |
|------------------------|-----------|-------------------------------------------|--------------------------------|-----------------------------|-----------------------------|-----------------------------|
| Types of Instructions | Mnemonics | Operand | Length (Number of Bytes) | issue | repeat | latency |
| Branch instructions | Bcond | disp9 | 2 | 2 to 6 ^{*3} | 2 to 6 ^{*3} | 2 to 6 ^{*3} |
| | | disp9 (When Branch prediction is matched) | 2 | 2 to 3 ^{*3} | 2 to 3 ^{*3} | 2 to 3 ^{*3} |
| | | disp17 | 4 | 2 to 6 ^{*3} | 2 to 6 ^{*3} | 2 to 6 ^{*3} |
| | JARL | disp22, reg2 | 4 | 2 to 3 ^{*3} | 2 to 3 ^{*3} | 2 to 3 ^{*3} |
| | | disp32, reg1 | 6 | 2 to 3 ^{*3} | 2 to 3 ^{*3} | 2 to 3 ^{*3} |
| | | [reg1], reg3 | 4 | 2 to 6 ^{*3} | 2 to 6 ^{*3} | 2 to 6 ^{*3} |
| | JMP | [reg1] | 2 | 2 to 6 ^{*3} | 2 to 6 ^{*3} | 2 to 6 ^{*3} |
| | | disp32 [reg1] | 6 | 2 to 7 ^{*3} | 2 to 7 ^{*3} | 2 to 7 ^{*3} |
| | JR | disp22 | 4 | 2 to 3 ^{*3} | 2 to 3 ^{*3} | 2 to 3 ^{*3} |
| | | disp32 | 6 | 2 to 3 ^{*3} | 2 to 3 ^{*3} | 2 to 3 ^{*3} |
| Loop instruction | LOOP | reg1, disp16 | 4 | 2 to 6 ^{*3} | 2 to 6 ^{*3} | 2 to 6 ^{*3} |
| Bit manipulation | CLR1 | bit#3, disp16 [reg1] | 4 | 1 | 1 | 4*4 |
| instruction | | reg2, [reg1] | 4 | 1 | 1 | 4*4 |
| | NOT1 | bit#3, disp16 [reg1] | 4 | 1 | 1 | 4*4 |
| | | reg2, [reg1] | 4 | 1 | 1 | 4*4 |
| | SET1 | bit#3, disp16 [reg1] | 4 | 1 | 1 | 4*4 |
| | | reg2, [reg1] | 4 | 1 | 1 | 4*4 |
| | TST1 | bit#3, disp16 [reg1] | 4 | 1 | 1 | 4*4 |
| | | reg2, [reg1] | 4 | 1 | 1 | 4*4 |
| Special instruction | | | | | | |
| Table reference branch | SWITCH | reg1 | 2 | 11 to 18 ^{*3} | 11 to 18 ^{*3} | 11 to 18 ^{*3} |
| Sub routine call | CALLT | imm6 | 2 | 17 | 17 | 17 |
| | CTRET | - | 4 | 8 | 8 | 8 |
| System call exception | SYSCALL | vector8 | 4 | 17 | 17 | 17 |
| Software exception | FETRAP | vector4 | 2 | 8 | 8 | 8 |
| | TRAP | vector5 | 4 | 8 | 8 | 8 |
| Return from | EIRET | - | 4 | 8 | 8 | 8 |
| exception processing | FERET | — | 4 | 8 | 8 | 8 |
| El level interrupt | DI | — | 4 | 3 | 3 | 3 |
| | EI | — | 4 | 3 | 3 | 3 |
| Restoration from & | DISPOSE | imm5, list12 | 4 | N+1 ^{*5} | N+2 ^{*5} | N+1 ^{*5} |
| Storage on stack | | imm5, list12, [reg1] | 4 | N+3 to N+8 ^{*5} | N+4 to N+8 ^{*5} | N+3 to N+8 ^{*5} |
| | PREPARE | list12, imm5 | 4 | N+1 ^{*5} | N+2 ^{*5} | N+1 ^{*5} |
| | | list12, imm5, sp | 4 | N+2 ^{*5} | N+3 ^{*5} | N+2 ^{*5} |
| | | list12, imm5, imm16 | 6 | N+2 ^{*5} | N+3 ^{*5} | N+2 ^{*5} |
| | | list12, imm5, imm16<<16 | 6 | N+2 ^{*5} | N+3 ^{*5} | N+2 ^{*5} |
| | | list12, imm5, imm32 | 8 | N+2 ^{*5} | N+3 ^{*5} | N+2 ^{*5} |
| | POPSP | rh-rt | 4 | N+1 ^{*6} | N+2 ^{*6} | N+1 ^{*6} |
| | PUSHSP | rh-rt | 4 | N+1 ^{*6} | N+2 ^{*6} | N+1 ^{*6} |



APPENDIX B NUMBER OF INSTRUCTION EXECTUION CLOCKS

| | | | | Instruction | Number | of Executio | n Clocks |
|----|---------------------|-----------|--------------------|--------------------------------|-----------------|-----------------|-----------------|
| Ту | pes of Instructions | Mnemonics | Operand | Length (Number of Bytes) | issue | repeat | latency |
| | System register | LDSR | reg2, regID, selID | 4 | 3 ^{*7} | 3 ^{*7} | 3 |
| | operation | STSR | regID, reg2, selID | 4 | 1 | 1 | 3 |
| | Exclusive control | CAXI | [reg1], reg2, reg3 | 4 | 1 | 1 | 8 ^{*4} |
| | | LDL.W | [reg1], reg3 | 4 | 1 | 1 | 3 ^{*1} |
| | | STC.W | reg3, [reg1] | 4 | 1 | 1 | 6 ^{*4} |
| | Stop | HALT | — | 4 | 1 | 1 | 1 |
| | | SNOOZE | — | 4 | *8 | *8 | *8 |
| | Synchronization | SYNCE | — | 2 | 1 | 1 | 1 |
| | | SYNCI | — | 2 | *9 | *9 | *9 |
| | | SYNCM | — | 2 | *10 | *10 | *10 |
| | | SYNCP | — | 2 | *11 | *11 | *11 |
| | Others | NOP | _ | 2 | 1 | 1 | 1 |
| | | RIE | — | 4 | 8 | 8 | 8 |

(2) Cache instruction

| | | | Instruction | Number of Execution Clocks | | |
|-----------------------------|-----------|-----------------|--------------------------------|----------------------------|------------------|------------------|
| Type of Instructions | Mnemonics | Operand | Length (Number of Bytes) | issue | repeat | latency |
| Cache operation instruction | CACHE | cacheop, [reg1] | 4 | 1 ^{*12} | 1 ^{*12} | 1 ^{*12} |
| Pre-fetch instruction | PREF | prefop, [reg1] | 4 | 1 ^{*12} | 1 ^{*12} | 1 ^{*12} |



(3) Floating-point operation instruction — single precision —

| | Manageria | Onesend | Instruction Length (Number of | Number of Execution Clocks | | |
|----------------------------|------------|------------------|-------------------------------------|----------------------------|--------|-------------------|
| Type of Instructions | | | | issuo | ropost | latonov |
| Floating-point arithmetic | ABSES | | 4 | 1 | 1 | 4 ^{*13} |
| operation | ADDF.S | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| | NEGF.S | rea2. rea3 | 4 | 1 | 1 | 4 ^{*13} |
| | SUBF.S | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| Floating-point | MULF.S | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| Multiply-accumulate/ | FMAES | ren1 ren2 ren3 | 4 | 1 | 1 | *13 |
| subtract operation | FMSES | | 4 | 1 | 1 | ч |
| | | | т А | 1 | 1 | ч |
| | ENMSES | | 4 | 1 | 1 | ч 4*13 |
| Floating-point subtraction | DIVES | | - - | ' 8*14 | 8 | 11*13 |
| Square root of a | | | - л | o*14 | 0 | 11*13 |
| Floatingpoint | RECIFF.3 | | 4 | o 01*14 | 0 | 04*13 |
| value /Reciprocal | ROQRIF.S | | 4 | 21 1 4*14 | 21 | 24 17*13 |
| | SURTE.S | | 4 | 14 | 14 | 17 4*13 |
| floatingpoint formats/ | | reg2, reg3 | 4 | 1 | 1 | 4 [*] 13 |
| Conversion between | CVTF.LS | reg2, reg3 | 4 | 1 | 1 | 4 *13 |
| point formats | | | 4 | 1 | 1 | 4 [*] 13 |
| | CVTF.SL | | 4 | 1 | 1 | 4*13 |
| | CVTF.SUL | | 4 | 1 | 1 | 4*13 |
| | | reg2, reg3 | 4 | 1 | 1 | 4 ¹⁰ |
| | CVTF.SW | reg2, reg3 | 4 | 1 | 1 | 4 *13 |
| | CVTF.ULS | reg2, reg3 | 4 | 1 | 1 | 4 *13 |
| | CVTF.UWS | reg2, reg3 | 4 | 1 | 1 | 4 10 |
| | CVIF.WS | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | CEILF.SL | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | CEILF.SUL | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | CEILF.SUW | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | CEILF.SW | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | FLOORF.SL | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | FLOORF.SUL | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | FLOORF.SUW | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | FLOORF.SW | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | TRNCF.SL | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | TRNCF.SUL | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | TRNCF.SUW | reg2, reg3 | 4 | 1 | 1 | 4 13 |
| | TRNCF.SW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |



| Type of Instructions | | | | Instruction Length (Number of Bytes) | Number of Execution Clocks | | |
|-------------------------------------------|------------------------------|------------------|----------------------|-----------------------------------------------|----------------------------|------------------|------------------|
| | | Mnemonics | Operand | | issue | repeat | latency |
| Flo | pating-point mparison | CMPF.S | cond, reg1, reg2, cc | 4 | 1 | 1 | 1 |
| | Transfer with conditions | CMOVF.S | cc, reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| | Bit transfer with conditions | TRFSR | сс | 4 | 1 | 1 | 5 |
| Floating-point maximum/ minimum values | MAXF.S | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} | |
| | MINF.S | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} | |

(4) Floating-point operation instruction — double precision —

| | P P P P P P P P P P P P P P P P P P P | | Instruction | Number of Execution Clocks | | |
|---------------------------------------------------------|---------------------------------------|------------------|--------------------------------|----------------------------|--------|-------------------|
| Type of Instructions | Mnemonics | Operand | Length (Number of Bytes) | issue | repeat | latency |
| Floating-point arithmetic | ABSF.D | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| operation | ADDF.D | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| | NEGF.D | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| | SUBF.D | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| Floating-point multiplication | MULF.D | reg1, reg2, reg3 | 4 | 4 | 4 | 7 ^{*13} |
| Floating-point division | DIVF.D | reg1, reg2, reg3 | 4 | 16 ^{*14} | 16 | 19 ^{*13} |
| Square root of a Floatingpoint value / Reciprocal | RECIPF.D | reg2, reg3 | 4 | 16 ^{*14} | 16 | 19 ^{*13} |
| | RSQRTF.D | reg2, reg3 | 4 | 45 ^{*14} | 45 | 48 ^{*13} |
| | SQRTF.D | reg2, reg3 | 4 | 30 ^{*14} | 30 | 33 ^{*13} |



APPENDIX B NUMBER OF INSTRUCTION EXECTUION CLOCKS

| Type of Instructions Mnemonics Operand Length of Bytes) issue repeat latency Conversion between fixedpoint and floating point formats/ CVTF.DL reg2, reg3 4 1 1 4 ¹¹³ CVTF.DL reg2, reg3 4 1 1 4 ¹¹³ CVTF.DUL reg2, reg3 4 1 1 4 ¹¹³ CVTF.DUL reg2, reg3 4 1 1 4 ¹¹³ CVTF.DUL reg2, reg3 4 1 1 4 ¹¹³ CVTF.DU reg2, reg3 4 1 1 4 ¹¹³ CVTF.DU reg2, reg3 4 1 1 4 ¹¹³ CVTF.UD reg2, reg3 4 1 1 4 ¹¹³ CEILF.DU reg2, reg3 | | Mnemonics Operand | | Instruction | Number of Execution Clocks | | |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------|-------------------|----------------------|--------------------------------|----------------------------|--------|------------------|
| Conversion between floatingpoint formats/ COVFF.DL reg2, reg3 4 1 1 4*13 CVTF.DS reg2, reg3 4 1 1 4*13 CVTF.DUL reg2, reg3 4 1 1 4*13 CVTF.DUL reg2, reg3 4 1 1 4*13 CVTF.DUW reg2, reg3 4 1 1 4*13 CVTF.DUW reg2, reg3 4 1 1 4*13 CVTF.DUW reg2, reg3 4 1 1 4*13 CVTF.DU reg2, reg3 4 1 1 4*13 CVTF.WD reg2, reg3 4 1 1 4*13 CEILF.DU reg2, reg3 4 1 1 4*13 CEILF.DU <th>Type of Instructions</th> <th>Operand</th> <th>Length (Number of Bytes)</th> <th>issue</th> <th>repeat</th> <th>latency</th> | Type of Instructions | | Operand | Length (Number of Bytes) | issue | repeat | latency |
| floatingpoint formats/ Conversion between fixedpoint and floating point formats CVTF.DS reg2, reg3 4 1 1 4 ⁺¹³ CVTF.DUL reg2, reg3 4 1 1 4 ⁺¹³ CVTF.DUW reg2, reg3 4 1 1 4 ⁺¹³ CVTF.DUW reg2, reg3 4 1 1 4 ⁺¹³ CVTF.DW reg2, reg3 4 1 1 4 ⁺¹³ CVTF.DW reg2, reg3 4 1 1 4 ⁺¹³ CVTF.UD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.UD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.UWD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.WD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.WD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.WD reg2, reg3 4 1 1 4 ⁺¹³ CEILF.DU reg2, reg3 4 1 1 4 ⁺¹³ CEILF.DU reg2, reg3 4 1 <t< td=""><td>Conversion between</td><td>CVTF.DL</td><td>reg2, reg3</td><td>4</td><td>1</td><td>1</td><td>4^{*13}</td></t<> | Conversion between | CVTF.DL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| fixedpoint and floating point formats CVTF.DUL reg2, reg3 4 1 1 4 ^{*13} CVTF.DUW reg2, reg3 4 1 1 4 ^{*13} CVTF.DUW reg2, reg3 4 1 1 4 ^{*13} CVTF.DW reg2, reg3 4 1 1 4 ^{*13} CVTF.LD reg2, reg3 4 1 1 4 ^{*13} CVTF.ULD reg2, reg3 4 1 1 4 ^{*13} CVTF.ULD reg2, reg3 4 1 1 4 ^{*13} CVTF.UVD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CEILF.DL reg2, reg3 4 1 1 4 ^{*13} CEILF.DUW reg2, reg3 4 1 1 4 ^{*13} CEILF.DUW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DL reg2, reg3 4 1 1 4 ^{*13} <td>floatingpoint formats/ Conversion between</td> <td>CVTF.DS</td> <td>reg2, reg3</td> <td>4</td> <td>1</td> <td>1</td> <td>4^{*13}</td> | floatingpoint formats/ Conversion between | CVTF.DS | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| DOINT ROTITION reg2, reg3 4 1 1 4 ^{*13} CVTF.DW reg2, reg3 4 1 1 4 ^{*13} CVTF.DW reg2, reg3 4 1 1 4 ^{*13} CVTF.DD reg2, reg3 4 1 1 4 ^{*13} CVTF.DD reg2, reg3 4 1 1 4 ^{*13} CVTF.UD reg2, reg3 4 1 1 4 ^{*13} CVTF.UD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CEILF.DL reg2, reg3 4 1 1 4 ^{*13} CEILF.DW reg2, reg3 4 1 1 4 ^{*13} CEILF.DW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DU reg2, reg3 4 1 1 4 ^{*13} FLOORF.DW | fixedpoint and floating | CVTF.DUL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CVTF.DW reg2, reg3 4 1 1 4 ^{*13} CVTF.LD reg2, reg3 4 1 1 4 ^{*13} CVTF.SD reg2, reg3 4 1 1 4 ^{*13} CVTF.ULD reg2, reg3 4 1 1 4 ^{*13} CVTF.ULD reg2, reg3 4 1 1 4 ^{*13} CVTF.UWD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CEILF.DL reg2, reg3 4 1 1 4 ^{*13} CEILF.DUL reg2, reg3 4 1 1 4 ^{*13} CEILF.DW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DL reg2, reg3 4 1 1 4 ^{*13} FLOORF.DL reg2, reg3 4 1 1 4 ^{*13} FLOORF.DW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DW reg | point iornais | CVTF.DUW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CVTF.LD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.SD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.ULD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.ULD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.UWD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.WD reg2, reg3 4 1 1 4 ⁺¹³ CVTF.WD reg2, reg3 4 1 1 4 ⁺¹³ CEILF.DL reg2, reg3 4 1 1 4 ⁺¹³ CEILF.DUW reg2, reg3 4 1 1 4 ⁺¹³ CEILF.DW reg2, reg3 4 1 1 4 ⁺¹³ FLOORF.DL reg2, reg3 4 1 1 4 ⁺¹³ FLOORF.DW reg2, reg3 4 1 1 4 ⁺¹³ FLOORF.DW reg2, reg3 4 1 1 4 ⁺¹³ TRANCF.DL reg2, reg3 4 1 1 4 ⁺¹³ TRANCF.DUW | | CVTF.DW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CVTF.SD reg2, reg3 4 1 1 4 ^{*13} CVTF.ULD reg2, reg3 4 1 1 4 ^{*13} CVTF.UWD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CVTF.WD reg2, reg3 4 1 1 4 ^{*13} CEILF.DL reg2, reg3 4 1 1 4 ^{*13} CEILF.DU reg2, reg3 4 1 1 4 ^{*13} CEILF.DU reg2, reg3 4 1 1 4 ^{*13} CEILF.DUW reg2, reg3 4 1 1 4 ^{*13} CEILF.DW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DL reg2, reg3 4 1 1 4 ^{*13} FLOORF.DUW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DUW reg2, reg3 4 1 1 4 ^{*13} FLOORF.DUW reg2, reg3 4 1 1 4 ^{*13} TRANCF.DU < | | CVTF.LD | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CVTF.ULD reg2, reg3 4 1 1 4*13 CVTF.UWD reg2, reg3 4 1 1 4*13 CVTF.WD reg2, reg3 4 1 1 4*13 CEILF.DL reg2, reg3 4 1 1 4*13 CEILF.DL reg2, reg3 4 1 1 4*13 CEILF.DUL reg2, reg3 4 1 1 4*13 CEILF.DUW reg2, reg3 4 1 1 4*13 CEILF.DUW reg2, reg3 4 1 1 4*13 CEILF.DW reg2, reg3 4 1 1 4*13 FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DU reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 | | CVTF.SD | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CVTF.UWD reg2, reg3 4 1 1 4*13 CVTF.WD reg2, reg3 4 1 1 4*13 CEILF.DL reg2, reg3 4 1 1 4*13 CEILF.DL reg2, reg3 4 1 1 4*13 CEILF.DUL reg2, reg3 4 1 1 4*13 CEILF.DUW reg2, reg3 4 1 1 4*13 CEILF.DW reg2, reg3 4 1 1 4*13 CEILF.DW reg2, reg3 4 1 1 4*13 FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 | | CVTF.ULD | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CVTF.WDreg2, reg34114*13CEILF.DLreg2, reg34114*13CEILF.DULreg2, reg34114*13CEILF.DUWreg2, reg34114*13CEILF.DWreg2, reg34114*13CEILF.DWreg2, reg34114*13FLOORF.DLreg2, reg34114*13FLOORF.DULreg2, reg34114*13FLOORF.DUWreg2, reg34114*13FLOORF.DUWreg2, reg34114*13FLOORF.DUreg2, reg34114*13TRANCF.DLreg2, reg34114*13TRANCF.DUUreg2, reg34114*13TRANCF.DUWreg2, reg34114*13TRANCF.DUWreg2, reg34114*13TRANCF.DUWreg2, reg34114*13TRANCF.DUWreg2, reg34114*13 | | CVTF.UWD | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CEILF.DL reg2, reg3 4 1 1 4* ¹³ CEILF.DUL reg2, reg3 4 1 1 4* ¹³ CEILF.DUW reg2, reg3 4 1 1 4* ¹³ CEILF.DUW reg2, reg3 4 1 1 4* ¹³ CEILF.DW reg2, reg3 4 1 1 4* ¹³ CEILF.DW reg2, reg3 4 1 1 4* ¹³ FLOORF.DL reg2, reg3 4 1 1 4* ¹³ FLOORF.DUL reg2, reg3 4 1 1 4* ¹³ FLOORF.DUW reg2, reg3 4 1 1 4* ¹³ FLOORF.DUW reg2, reg3 4 1 1 4* ¹³ FLOORF.DW reg2, reg3 4 1 1 4* ¹³ TRANCF.DL reg2, reg3 4 1 1 4* ¹³ TRANCF.DUW reg2, reg3 4 1 1 4* ¹³ TRANCF.DUW reg2, reg3 4 1 1 4* ¹³ TRANCF.DUW< | | CVTF.WD | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CEILF.DUL reg2, reg3 4 1 1 4*13 CEILF.DUW reg2, reg3 4 1 1 4*13 CEILF.DW reg2, reg3 4 1 1 4*13 CEILF.DW reg2, reg3 4 1 1 4*13 FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DU reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 </td <td></td> <td>CEILF.DL</td> <td>reg2, reg3</td> <td>4</td> <td>1</td> <td>1</td> <td>4^{*13}</td> | | CEILF.DL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CEILF.DUW reg2, reg3 4 1 1 4*13 CEILF.DW reg2, reg3 4 1 1 4*13 FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, re | | CEILF.DUL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| CEILF.DW reg2, reg3 4 1 1 4*13 FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | CEILF.DUW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| FLOORF.DL reg2, reg3 4 1 1 4*13 FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 | | CEILF.DW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| FLOORF.DUL reg2, reg3 4 1 1 4*13 FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | FLOORF.DL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| FLOORF.DUW reg2, reg3 4 1 1 4*13 FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | FLOORF.DUL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| FLOORF.DW reg2, reg3 4 1 1 4*13 TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | FLOORF.DUW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| TRANCF.DL reg2, reg3 4 1 1 4*13 TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | FLOORF.DW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| TRANCF.DUL reg2, reg3 4 1 1 4*13 TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | TRANCF.DL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| TRANCF.DUW reg2, reg3 4 1 1 4*13 TRANCF.DW reg2, reg3 4 1 1 4*13 | | TRANCF.DUL | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| TRANCF.DW reg2, reg3 4 1 1 4*13 | | TRANCF.DUW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| | | TRANCF.DW | reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| Floating-point comparison CMPF.D cond, reg1, reg2, cc 4 1 1 1 | Floating-point comparison | CMPF.D | cond, reg1, reg2, cc | 4 | 1 | 1 | 1 |
| Transfer with conditionsCMOVF.Dcc, reg1, reg2, reg34114*13 | Transfer with conditions | CMOVF.D | cc, reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| Floating-point maximum/ MAXF.D reg1, reg2, reg3 4 1 1 4*13 | Floating-point maximum/ | MAXF.D | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |
| MINF.D reg1, reg2, reg3 4 1 1 4*13 | minimum values | MINF.D | reg1, reg2, reg3 | 4 | 1 | 1 | 4 ^{*13} |

Note 1. When no waiting is required (3 + number of wait states for read access).

Note 2. N = int(((number of valid bits in absolute value of dividend) – (number of valid bits in absolute value of divisor)) ÷ 2) + 1

N becomes 1 for the result of N < 1. Division by 0 leads to N being 0. The range of N is from 0 to 16. Note 3. Executing an instruction to rewrite the contents of the PSW register immediately beforehand does not affect the number of clock number of clock numbers of the numbers of th

the number of clock cycles for execution. Even if an immediately preceding instruction has rewritten the contents of the PSW register, parallel execution is possible.

Note 4. When no waiting is required (4 + number of wait states for read access).

 Note 5. "N" depends on the total number of registers specified by list12, but not on the register numbers. Since up to two registers are handled per clock cycle, if no waiting is required, the values are as shown below.
PREPARE: Minimum value is 1, maximum value is 6 (if the EP register is updated, add 1 clock cycle).

DISPOSE: Minimum value is 1, maximum value is 6 (if accompanied by JMP, add 2 clock cycles).

Note 6. "N" depends on the total number of registers specified by (rh-rt).

Since up to two registers are handled per clock cycle, if no waiting is required, the values are as shown below.

PUSHSH: Minimum value is 1, maximum value is 16.

POPSP: Minimum value is 1, maximum value is 16.

Note 7. When accessing the system register to control the operation of PSW or so, stop issuing the subsequent

RENESAS

instructions.

If not, perform the operation with issue = 1.

- Note 8. The number of execution clocks for the SNOOZE instruction is specified by the hardware specifications. For details, see the hardware manual of the corresponding product.
- Note 9. Wait for the hazard resolution for the instruction.
- Note 10. Perform the synchronization of memory access.
- Note 11. Wait for the synchronization of the pipeline.
- Note 12. Though the execution of the instruction is completed, the completion of the internal processing depends on the state of the instruction fetch unit.
- Note 13. "latency" might be added by 1 depending on the subsequent instruction. For details, see Note 1.
- Note 14. "issue" is set to 1 for the instructions other than the one with the floating-point division (DIVF, RECIPF, RSQRTF, or SQRTF)

Note 1. Example of execution clocks

| Symbol | Description |
|---------|---------------------------------------------------------------------------------------------------|
| issue | When the other instruction is executed immediately after the execution of the current instruction |
| repeat | When the same instruction is repeated immediately after the execution of the current instruction |
| latency | When the following instruction uses the result of the current instruction*1 |
| | |

Note 1. In the following case, "latency" is added by 1.

• When the preceding instruction is the floating-point operation instruction, the instruction other than the floating-point operation one follows.

• When the subsequent instruction is the floating-point operation instruction, the instruction other than the load, store, bit-manipulation, special (memory access), or floating-point operation one precedes.



APPENDIX C REGISTER INDEX

APPENDIX C REGISTER INDEX

А

| ASID | 49 |
|--------------------------------|----|
| С | |
| CDBCR CTBP CTPC CTPSW | |
| E | |
| EBASE | 53 |
| EIIC | 46 |
| EIPC | 40 |
| EIPSW | 41 |
| EIWR | 49 |

F

| FEIC | 47 |
|-------|----|
| FEPC | 42 |
| FEPSW | 43 |
| FEWR | 49 |
| FPCC | 67 |
| FPCFG | 68 |
| FPEPC | 66 |
| FPSR | 63 |
| FPST | 66 |

Н

| HTCFG0 | |
|--------|--|
| | |

I

| ICCFG | |
|--------|----|
| ICCTRL | 80 |
| ICDATH | 80 |
| ICDATL | 79 |
| ICERR | |
| ICSR | 60 |
| ICTAGH | 79 |
| ICTAGL | 78 |
| INTBP | 54 |
| INTCFG | 61 |
| ISPR | 59 |
| | |

Μ

| 73 |
|----|
| 73 |
| 57 |
| 74 |
| 73 |
| 57 |
| 50 |
| 51 |
| |
| |

| MPBRGN MPLAn MPM MPRC MPTRGN MPUAn P | 72 75 71 72 72 75 |
|--------------------------------------------------------|----------------------------------|
| PC | 38 |
| PID | 55 |
| PMR | 60 |
| PSW | 44 |
| R | |
| RBASE | 53 |
| S | |
| | |

| SCBP | 56 |
|-------|----|
| SCCFG | 56 |

APPENDIX D INSTRUCTION INDEX

LD.W207

INSTRUCTION INDEX APPENDIX D

А

| ABSF.D | 313 |
|--------|-----|
| ABSF.S | 314 |
| ADD | 152 |
| ADDF.D | 315 |
| ADDF.S | 317 |
| ADDI | 153 |
| ADF | 154 |
| AND | 155 |
| ANDI | 156 |

В

| d1 | 57 |
|----|-------------------------|
| | 60 |
| | 61 |
| | 62 |
| 0 | d 1 1 1 1 1 |

С

| CACHE | 301 |
|-----------|-----|
| CALLT | 163 |
| CAXI | 165 |
| CEILF.DL | 319 |
| CEILF.DUL | 320 |
| CEILF.DUW | 321 |
| CEILF.DW | 322 |
| CEILF.SL | 323 |
| CEILF.SUL | 324 |
| CEILF.SUW | 325 |
| CEILF.SW | 326 |
| CLL | 167 |
| CLR1 | 168 |
| CMOV | 170 |
| CMOVF.D | 327 |
| CMOVF.S | 328 |
| CMP | 172 |
| CMPF.D | 329 |
| CMPF.S | 333 |
| CTRET | 173 |
| CVTF.DL | 337 |
| CVTF.DS | 338 |
| CVTF.DUL | 339 |
| CVTF.DUW | 340 |
| CVTF.DW | 341 |
| CVTF.HS | 342 |
| CVTF.LD | 343 |
| CVTF.LS | 344 |
| CVTF.SD | 345 |
| CVTF.SH | 347 |
| CVTF.SL | 346 |
| CVTF.SUL | 348 |
| CVTF.SUW | 349 |
| CVTF.SW | 350 |
| CVTF.ULD | 351 |
| CVTF.ULS | 352 |

| EI | 186 |
|-------|---------|
| EIRET | 187 |

F

| FERET 1 | 88 |
|--------------|-----|
| FETRAP 1 | 89 |
| FLOORF.DL3 | 61 |
| FLOORF.DUL 3 | 62 |
| FLOORF.DUW | 63 |
| FLOORF.DW | 64 |
| FLOORF.SL | 65 |
| FLOORF.SUL | 666 |
| FLOORF.SUW3 | 67 |
| FLOORF.SW | 868 |
| FMAF.S | 69 |
| FMSF.S | 371 |
| FNMAF.S 3 | 373 |
| FNMSF.S 3 | 375 |

Н

| HALT | . 191 |
|------|-------|
| HSH | . 193 |
| HSW | . 194 |

J

| JARL JMP JR L | 195 197 198 |
|------------------------|-------------------|
| LD.B | 199 |
| LD.BU | 200 |
| LD.DW | 202 |

| LD.B | 199 |
|-------|-----|
| LD.BU | 200 |
| LD.DW | 202 |
| LD.H | 203 |
| LD.HU | 205 |

| LDSR210 LOOP211 | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Μ | |
| MAC 213 MACU 214 MAXF.D 377 MAXF.S 379 MINF.D 380 MINF.S 382 MOV 215 MOVEA 217 MOVHI 218 MUL 219 MULF.D 385 MULF.S 386 MULH 220 MULHI 223 | |
| NEGF.D | |
| OR228 ORI229 P | |
| POPSP | |
| RECIPF.D | |
| SAR239 SASF241 SATADD242 | |



SATSUB244

SATSUBI246

| SATSUBR | .247 | Х |
|---------|-------|----|
| SBF | .248 | |
| SCH0L | .249 | X |
| SCHOR | .250 | X |
| SCH1L | .251 | 7 |
| SCH1R | . 252 | 2 |
| SET1 | .253 | 7 |
| SETF | .255 | 7 |
| SHL | .257 | ~/ |
| SHR | .259 | |
| SLD.B | .261 | |
| SLD.BU | .262 | |
| SID H | 263 | |
| SI D HU | 264 | |
| SI D W | 265 | |
| SN007E | 266 | |
| SORTED | 393 | |
| SORTES | 304 | |
| SST B | 268 | |
| 991.D | 260 | |
| SSTW | 270 | |
| ST B | 270 | |
| ST.D.W/ | 277 | |
| ST.DW | .272 | |
| ST.W | 275 | |
| STC W | .275 | |
| 910.W | .270 | |
| STOR | 280 | |
| SUBED | 305 | |
| | 207 | |
| | 201 | |
| | .201 | |
| | . 202 | |
| 5XB | .283 | |
| SXH | .284 | |
| SYNCE | .285 | |
| SYNCI | .286 | |
| SYNCM | .287 | |
| SYNCP | .288 | |
| SYSCALL | .289 | |
| | | |

| XOR | 296 |
|------|-----|
| XORI | 297 |
| Z | |
| ZXB | 298 |
| ZXH | 299 |

| TRAP | 291 |
|-----------|-----|
| TRFSR | 398 |
| TRNCF.DL | 399 |
| TRNCF.DUL | 400 |
| TRNCF.DUW | 401 |
| TRNCF.DW | 402 |
| TRNCF.SL | 403 |
| TRNCF.SUL | 404 |
| TRNCF.SUW | 405 |
| TRNCF.SW | 406 |
| TST | 293 |
| TST1 | 294 |
| | |

REVISION HISTORY

RH850G3MH User's Manual: Software

 Page
 Description
 Classification

 - First Edition issued
 -

Note: The classification in the table above means as follows.

(a): Error correction (b): Specifications added or changed (c): descriptions or notes added or changed



| RH850G3MH | User's Manual: Software |
|-------------------|---------------------------------|
| Publication Date: | Rev.1.00 Mar 05, 2015 |
| Published by: | Renesas Electronics Corporation |



SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information. California Eastern Laboratories, Inc. 4590 Patrick Henry Drive, Santa Clara, California 95054-1817, U.S.A. Tei: +1-408-919-2500, Fax: +1-408-988-0279 Renesas Electronics Europe Limited Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K Tei: +44-1628-585-100, Fax: +44-1628-585-900 Renesas Electronics Europe GmbH Arcadiastrasse 10, 40472 Düsseldorf, Germany Tei: +49-211-6503-0, Fax: +44-1628-585-900 Renesas Electronics Chinal Co., Ltd. Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China Tei: +49-2155, Fax: +86-10-8235-7679 Renesas Electronics (Shanghai) Co., Ltd. Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333 Tei: +86-21-2226-088, Fax: +852-2286-9002 Renesas Electronics (Shanghai) Co., Ltd. Unit 101-111, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong Tei: +852-2455-6688, Fax: +852 2486-9022 Renesas Electronics Taiwan Co., Ltd. 137, No. 363, Fu Shing North Road, Taipei 10543, Taiwan Tei: +886-24175-9800, Fax: +886 2-8175-9670 Renesas Electronics Singapore Pte. Ltd. 80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949 Tei: +65-213-0200, Fax: +856 2-513-0300 Renesas Electronics Indiaysia Sdn.Bhd. Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, JIn Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia Tei: +80-3-735-9390, Fax: +856-256-5501 Renesas Electronics India Pvt. Ltd. No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India Tei: +10-0-75020700, Fax: +918-05-020777 Renesas Electronics Korea Co., Ltd. No.777C, 100 Feet Road, HAL II Stage, Indiranagar, Bangalore, India Tei: +41-10-0 Feet Road, HAL II Stage, Indiranagar, Bangalore, India Tei: +41-10-0 Feet Road, HAL II Stage, Indiranagar, Bangalore, India Tei: +41-10-0 Feet Road, HAL II Stage, Indiranagar, Bangalore, India Tei: +41-10-0 Feet

> © 2015 Renesas Electronics Corporation. All rights reserved. Colophon 4.0

RH850G3MH

