<Arithmetic instruction>

ADD	Add register/immediate Add

[Instruction format] (1) ADD reg1, reg2

(2) ADD imm5, reg2

[Operation] (1) GR [reg2] ← GR [reg2] + GR [reg1]

(2) GR [reg2] ← GR [reg2] + sign-extend (imm5)

[Format] (1) Format I

(2) Format II

[Opcode]

	15	0
(1)	rrrrr001110RRRRR	

	15	0
(2)	rrrrr010010iiiii	

[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.

<Branch instruction>

Bcond	Branch on condition code with 9-bit displacement Conditional branch

[Instruction format] Bcond disp9

[Operation] if conditions are satisfied
then PC ← PC + sign-extend (disp9)

[Format] Format III

[Opcode] 15 0

ddddd1011dddcccc

ddddddddd is the higher 8 bits of disp9.
cccc is the condition code of the condition indicated by cond (refer to **Table 5-5 Bcond Instructions**).

[Flags] CY --
OV --
S --
Z --
SAT --

[Description] 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).

[Comment] 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.

Table 5-5. Bcond Instructions

Instruction		Condition Code (cccc)	Flag Status	Branch Condition
Signed integer	BGE	1110	(S xor OV) = 0	Greater than or equal to signed
	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 integer	BH	1011	(CY or Z) = 0	Higher (Greater than)
	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)
	BSA	1101	SAT = 1	Saturated
	BT	0010	Z = 1	True
	BV	0000	OV = 1	Overflow
	BZ	0010	Z = 1	Zero

Caution **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.**

<Arithmetic instruction>

CMP

Compare register/immediate (5-bit)

Compare

[Instruction format] (1) CMP reg1, reg2

(2) CMP imm5, reg2

[Operation] (1) result ← GR [reg2] – GR [reg1]

(2) result ← GR [reg2] – sign-extend (imm5)

[Format] (1) Format I

(2) Format II

[Opcode]

(1)

15															0
r	r	r	r	r	0	0	1	1	1	1	R	R	R	R	R

(2)

15															0
r	r	r	r	r	0	1	0	0	1	1	i	i	i	i	i

[Flags]

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	“1” if the operation result is “0”; otherwise, “0”.
SAT	--

[Description] (1) Compares the word data of general-purpose register reg2 with the word data of general-purpose register reg1 and outputs the result through the PSW flags. Comparison is performed by subtracting the reg1 contents from the reg2 word data. General-purpose registers reg1 and reg2 are not affected.

(2) Compares the word data of general-purpose register reg2 with the 5-bit immediate data, sign-extended to word length, and outputs the result through the PSW flags. Comparison is performed by subtracting the sign-extended immediate data from the reg2 word data. General-purpose register reg2 is not affected.

<Branch instruction>

JMP

Jump register

Unconditional branch (register relative)

[Instruction format] (1) JMP [reg1]

(2) JMP disp32 [reg1]

[Operation] (1) PC ← GR [reg1]

(2) PC ← GR [reg1] + disp32

[Format] (1) Format I

(2) Format VI

[Opcode]

(1)

15	0
00000000011RRRRR	

(2)

15	0	31	16	47	32
00000110111RRRRR		ddddddddddddddd0		DDDDDDDDDDDDDDDD	

DDDDDDDDDDDDDDDDddddddddddddddd is the higher 31 bits of disp32.

[Flags]

CY	--
OV	--
S	--
Z	--
SAT	--

[Description] (1) Transfers the control to the address specified by general-purpose register reg1. Bit 0 of the address is masked to “0”.

(2) 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”.

[Comment] Using this instruction as the subroutine control instruction requires the return PC to be specified by general-purpose register reg1.

<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)

GR [reg2] ← Load-memory (adr, Word)

(2) adr ← GR [reg1] + sign-extend (disp23)

GR [reg3] ← Load-memory (adr, Word)

[Format] (1) Format VII

(2) Format XIV

[Opcode]

(1)

15	0	31	16
rrrrr111001RRRRR		ddddddddddddddd1	

Where ddddddddddddddd is the higher 15 bits of disp16.

(2)

15	0	31	16	47	32
00000111100RRRRR		wwwwwdddddd01001		DDDDDDDDDDDDDDDD	

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] (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. 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.

<Arithmetic instruction>

MOV	Move register/immediate (5-bit) /immediate (32-bit) Data transfer

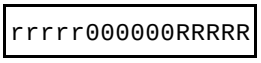
[Instruction format] (1) MOV reg1, reg2
(2) MOV imm5, reg2
(3) MOV imm32, reg1

[Operation] (1) GR [reg2] ← GR [reg1]
(2) GR [reg2] ← sign-extend (imm5)
(3) GR [reg1] ← imm32

[Format] (1) Format I
(2) Format II
(3) Format VI

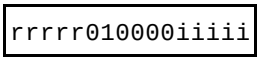
[Opcode]

(1)



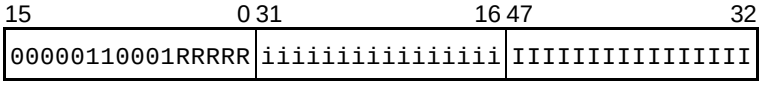
rrrrr ≠ 00000 (Do not specify r0 for reg2.)

(2)



rrrrr ≠ 00000 (Do not specify r0 for reg2.)

(3)



i (bits 31 to 16) refers to the lower 16 bits of 32-bit immediate data.
I (bits 47 to 32) refers to the higher 16 bits of 32-bit immediate data.

[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 general-purpose 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).**

<Multiply instruction>

MUL	Multiply word by register/immediate (9-bit) Multiplication of (signed) word data

[Instruction format] (1) MUL reg1, reg2, reg3

(2) MUL imm9, reg2, reg3

[Operation] (1) GR [reg3] || GR [reg2] ← GR [reg2] × GR [reg1]

(2) GR [reg3] || GR [reg2] ← GR [reg2] × sign-extend (imm9)

[Format] (1) Format XI

(2) Format XII

[Opcode]

	15	0	31	16
(1)	rrrrr111111RRRRR		wwwww01000100000	

	15	0	31	16
(2)	rrrrr111111iiiii		wwwww01001IIII00	

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.
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.

[Comment] 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.

<Special instruction>

NOP	No operation No operation

[Instruction format] NOP

[Operation] PC for this instruction is incremented by +2 (nothing else is done).

[Format] Format I

[Opcode]

15	0
0000000000000000	

[Flags]

CY	--
OV	--
S	--
Z	--
SAT	--

[Description] PC for this instruction is incremented by +2 (nothing else is done).

[Comment] The opcode is the same as that of MOV r0, r0.

<Store instruction>

ST.W	Store word Storage of word data

[Instruction format] (1) ST.W reg2, disp16 [reg1]

(2) ST.W reg3, disp23 [reg1]

[Operation] (1) adr ← GR [reg1] + sign-extend (disp16)

Store-memory (adr, GR [reg2], Word)

(2) adr ← GR [reg1] + sign-extend (disp23)

Store-memory (adr, GR [reg3], Word)

[Format] (1) Format VII

(2) Format XIV

[Opcode]

(1)

15	0	31	16
rrrrr111011RRRRR		ddddddddddddddd1	

Where ddddddddddddddd is the higher 15 bits of disp16.

(2)

15	0	31	16	47	32
00000111100RRRRR		wwwwwddddddd01111		DDDDDDDDDDDDDDDD	

Where RRRRR = reg1, wwwww = reg3.

ddddddd 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] (1) Adds the data of general-purpose register reg1 to the 16-bit displacement data, sign-extended to word length, to generate a 32-bit address and stores the word data of general-purpose register reg2 to the generated 32-bit address.

(2) Adds the data of general-purpose register reg1 to the 23-bit displacement data, sign-extended to word length, to generate a 32-bit address and stores the lowest word data of general-purpose register reg3 to the generated 32-bit address.

<Arithmetic instruction>

SUB	Subtract Subtraction

[Instruction format] SUB reg1, reg2

[Operation] GR [reg2] ← GR [reg2] – GR [reg1]

[Format] Format I

[Opcode]

15	0
rrrrr001101RRRRR	

[Flags]

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	“1” if the operation result is “0”; otherwise, “0”.
SAT	--

[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. General-purpose register reg1 is not affected.

<Arithmetic instruction>

SUBR	Subtract reverse Reverse subtraction

[Instruction format] SUBR reg1, reg2

[Operation] GR [reg2] ←GR [reg1] – GR [reg2]

[Format] Format I

[Opcode]

15															0
r	r	r	r	r	0	0	1	1	0	0	R	R	R	R	R

[Flags]

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	“1” if the operation result is “0”; otherwise, “0”.
SAT	--

[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. General-purpose register reg1 is not affected.

APPENDIX A LIST OF INSTRUCTIONS

A.1 Basic Instructions

Table A-1 shows an alphabetized list of basic instruction functions.

Table A-1. Basic Instruction Function List (Alphabetic Order) (1/4)

Mnemonic	Operand	Format	Flag					Function of Instruction
			CY	OV	S	Z	SAT	
ADD	reg1, reg2	I	0/1	0/1	0/1	0/1	–	Add
ADD	imm5, reg2	II	0/1	0/1	0/1	0/1	–	Add
ADDI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	–	Add
ADF	cccc, reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	–	Conditional add
AND	reg1, reg2	I	–	0	0/1	0/1	–	AND
ANDI	imm16, reg1, reg2	VI	–	0	0/1	0/1	–	AND
Bcond	disp9	III	–	–	–	–	–	Conditional branch
BSH	reg2, reg3	XII	0/1	0	0/1	0/1	–	Byte swap of halfword data
BSW	reg2, reg3	XII	0/1	0	0/1	0/1	–	Byte swap of word data
CALLT	imm6	II	–	–	–	–	–	Subroutine call with table look up
CAXI	[reg1], reg2, reg3	IX	0/1	0/1	0/1	0/1	–	Comparison and swap
CLR1	bit#3, disp16 [reg1]	VIII	–	–	–	0/1	–	Bit clear
CLR1	reg2, [reg1]	IX	–	–	–	0/1	–	Bit clear
CMOV	cccc, reg1, reg2, reg3	XI	–	–	–	–	–	Conditional transfer
CMOV	cccc, imm5, reg2, reg3	XII	–	–	–	–	–	Conditional transfer
CMP	reg1, reg2	I	0/1	0/1	0/1	0/1	–	Comparison
CMP	imm5, reg2	II	0/1	0/1	0/1	0/1	–	Comparison
CTRET	(None)	X	0/1	0/1	0/1	0/1	0/1	Return from subroutine call
DI	(None)	X	–	–	–	–	–	Disable EI level maskable exception
DISPOSE	imm5, list12	XIII	–	–	–	–	–	Stack frame deletion
DISPOSE	imm5, list12, [reg1]	XIII	–	–	–	–	–	Stack frame deletion
DIV	reg1, reg2, reg3	XI	–	0/1	0/1	0/1	–	Division of (signed) word data
DIVH	reg1, reg2	I	–	0/1	0/1	0/1	–	Division of (signed) halfword data
DIVH	reg1, reg2, reg3	XI	–	0/1	0/1	0/1	–	Division of (signed) halfword data.
DIVHU	reg1, reg2, reg3	XI	–	0/1	0/1	0/1	–	Division of (unsigned) halfword data
DIVQ	reg1, reg2, reg3	XI	–	0/1	0/1	0/1	–	Division of (signed) word data (variable steps)
DIVQU	reg1, reg2, reg3	XI	–	0/1	0/1	0/1	–	Division of (unsigned) word data (variable steps)
DIVU	reg1, reg2, reg3	XI	–	0/1	0/1	0/1	–	Division of (unsigned) word data

Table A-1. Basic Instruction Function List (Alphabetic Order) (2/4)

Mnemonic	Operand	Format	Flag					Function of Instruction
			CY	OV	S	Z	SAT	
EI	(None)	X	–	–	–	–	–	Enable EI level maskable exception
EIRET	(None)	X	0/1	0/1	0/1	0/1	0/1	Return from EI level exception
FERET	(None)	X	0/1	0/1	0/1	0/1	0/1	Return from FE level exception
FETRAP	vector	I	–	–	–	–	–	FE level software exception instruction
HALT	(None)	X	–	–	–	–	–	Halt
HSH	reg2, reg3	XII	0/1	0	0/1	0/1	–	Halfword swap of halfword data
HSW	reg2, reg3	XII	0/1	0	0/1	0/1	–	Halfword swap of word data
JARL	disp22, reg2	V	–	–	–	–	–	Branch and register link
JARL	disp32, reg1	VI	–	–	–	–	–	Branch and register link
JMP	[reg1]	I	–	–	–	–	–	Unconditional branch (register relative)
JMP	disp32 [reg1]	VI	–	–	–	–	–	Unconditional branch (register relative)
JR	disp22	V	–	–	–	–	–	Unconditional branch (PC relative)
JR	disp32	VI	–	–	–	–	–	Unconditional branch (PC relative)
LD.B	disp16 [reg1], reg2	VII	–	–	–	–	–	Load of (signed) byte data
LD.B	disp23 [reg1], reg3	XIV	–	–	–	–	–	Load of (signed) byte data
LD.BU	disp16 [reg1], reg2	VII	–	–	–	–	–	Load of (unsigned) byte data
LD.BU	disp23 [reg1], reg3	XIV	–	–	–	–	–	Load of (unsigned) byte data
LD.H	disp16 [reg1], reg2	VII	–	–	–	–	–	Load of (signed) halfword data
LD.H	disp23 [reg1], reg3	XIV	–	–	–	–	–	Load of (signed) halfword data
LD.HU	disp16 [reg1], reg2	VII	–	–	–	–	–	Load of (unsigned) halfword data
LD.HU	disp23 [reg1], reg3	XIV	–	–	–	–	–	Load of (unsigned) halfword data
LD.W	disp16 [reg1], reg2	VII	–	–	–	–	–	Load of word data
LD.W	disp23 [reg1], reg3	XIV	–	–	–	–		Load of word data
LDSR	reg2, regID	IX	–	–	–	–	–	Load to system register
MAC	reg1, reg2, reg3, reg4	XI	–	–	–	–	–	Multiply-accumulate for (signed) word data
MACU	reg1, reg2, reg3, reg4	XI	–	–	–	–	–	Multiply-accumulate for (unsigned) word data
MOV	reg1, reg2	I	–	–	–	–	–	Data transfer
MOV	imm5, reg2	II	–	–	–	–	–	Data transfer
MOV	imm32, reg1	VI	–	–	–	–	–	Data transfer
MOVEA	imm16, reg1, reg2	VI	–	–	–	–	–	Effective address transfer
MOVHI	imm16, reg1, reg2	VI	–	–	–	–	–	Higher halfword transfer
MUL	reg1, reg2, reg3	XI	–	–	–	–	–	Multiplication of (signed) word data
MUL	imm9, reg2, reg3	XII	–	–	–	–	–	Multiplication of (signed) word data
MULH	reg1, reg2	I	–	–	–	–	–	Multiplication of (signed) halfword data
MULH	imm5, reg2	II	–	–	–	–	–	Multiplication of (signed) halfword data
MULHI	imm16, reg1, reg2	VI	–	–	–	–	–	Multiplication of (signed) halfword immediate data

Table A-1. Basic Instruction Function List (Alphabetic Order) (3/4)

Mnemonic	Operand	Format	Flag					Function of Instruction
			CY	OV	S	Z	SAT	
MULU	reg1, reg2, reg3	XI	–	–	–	–	–	Multiplication of (unsigned) word data
MULU	imm9, reg2, reg3	XII	–	–	–	–	–	Multiplication of (unsigned) word data
NOP	(None)	I	–	–	–	–	–	Nothing else is done.
NOT	reg1, reg2	I	–	0	0/1	0/1	–	Logical negation (1's complement)
NOT1	bit#3, disp16 [reg1]	VIII	–	–	–	0/1	–	NOT bit
NOT1	reg2, [reg1]	IX	–	–	–	0/1	–	NOT bit
OR	reg1, reg2	I	–	0	0/1	0/1	–	OR
ORI	imm16, reg1, reg2	VI	–	0	0/1	0/1	–	OR immediate
PREPARE	list12, imm5	XIII	–	–	–	–	–	Create stack frame
PREPARE	list12, imm5, sp/imm	XIII	–	–	–	–	–	Create stack frame
RETI	(None)	X	0/1	0/1	0/1	0/1	0/1	Return from EI level software exception or interrupt
RIE	(None)	I/X	–	–	–	–	–	Reserved instruction exception
SAR	reg1, reg2	IX	0/1	0	0/1	0/1	–	Arithmetic right shift
SAR	imm5, reg2	II	0/1	0	0/1	0/1	–	Arithmetic right shift
SAR	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	–	Arithmetic right shift
SASF	cccc, reg2	IX	–	–	–	–	–	Shift and flag condition setting
SATADD	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	Saturated addition
SATADD	imm5, reg2	II	0/1	0/1	0/1	0/1	0/1	Saturated addition
SATADD	reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	0/1	Saturated addition
SATSUB	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	Saturated subtraction
SATSUB	reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	0/1	Saturated subtraction
SATSUBI	imm16, reg1, reg2	VI	0/1	0/1	0/1	0/1	0/1	Saturated subtraction
SATSUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	0/1	Saturated reverse subtraction
SBF	cccc, reg1, reg2, reg3	XI	0/1	0/1	0/1	0/1	–	Conditional subtraction
SCH0L	reg2, reg3	IX	0/1	0	0	0/1	–	Bit (0) search from MSB side
SCH0R	reg2, reg3	IX	0/1	0	0	0/1	–	Bit (0) search from LSB side
SCH1L	reg2, reg3	IX	0/1	0	0	0/1	–	Bit (1) search from MSB side
SCH1R	reg2, reg3	IX	0/1	0	0	0/1	–	Bit (1) search from LSB side
SET1	bit#3, disp16 [reg1]	VIII	–	–	–	0/1	–	Bit setting
SET1	reg2, [reg1]	IX	–	–	–	0/1	–	Bit setting
SETF	cccc, reg2	IX	–	–	–	–	–	Flag condition setting
SHL	reg1, reg2	IX	0/1	0	0/1	0/1	–	Logical left shift
SHL	imm5, reg2	II	0/1	0	0/1	0/1	–	Logical left shift
SHL	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	–	Logical left shift

Table A-1. Basic Instruction Function List (Alphabetic Order) (4/4)

Mnemonic	Operand	Format	Flag					Function of Instruction
			CY	OV	S	Z	SAT	
SHR	reg1, reg2	IX	0/1	0	0/1	0/1	–	Logical right shift
SHR	imm5, reg2	II	0/1	0	0/1	0/1	–	Logical right shift
SHR	reg1, reg2, reg3	XI	0/1	0	0/1	0/1	–	Logical right shift
SLD.B	disp7 [ep], reg2	IV	–	–	–	–	–	Load of (signed) byte data
SLD.BU	disp4 [ep], reg2	IV	–	–	–	–	–	Load of (unsigned) byte data
SLD.H	disp8 [ep], reg2	IV	–	–	–	–	–	Load of (signed) halfword data
SLD.HU	disp5 [ep], reg2	IV	–	–	–	–	–	Load of (unsigned) halfword data
SLD.W	disp8 [ep], reg2	IV	–	–	–	–	–	Load of word data
SST.B	reg2, disp7 [ep]	IV	–	–	–	–	–	Storage of byte data
SST.H	reg2, disp8 [ep]	IV	–	–	–	–	–	Storage of halfword data
SST.W	reg2, disp8 [ep]	IV	–	–	–	–	–	Storage of word data
ST.B	reg2, disp16 [reg1]	VII	–	–	–	–	–	Storage of byte data
ST.B	reg3, disp23 [reg1]	XIV	–	–	–	–	–	Storage of byte data
ST.H	reg2, disp16 [reg1]	VII	–	–	–	–	–	Storage of halfword data
ST.H	reg3, disp23 [reg1]	XIV	–	–	–	–	–	Storage of halfword data
ST.W	reg2, disp16 [reg1]	VII	–	–	–	–	–	Storage of word data
ST.W	reg3, disp23 [reg1]	XIV	–	–	–	–	–	Storage of word data
STSR	regID, reg2	IX	–	–	–	–	–	Storage of contents of system register
SUB	reg1, reg2	I	0/1	0/1	0/1	0/1	–	Subtraction
SUBR	reg1, reg2	I	0/1	0/1	0/1	0/1	–	Reverse subtraction
SWITCH	reg1	I	–	–	–	–	–	Jump with table look up
SXB	reg1	I	–	–	–	–	–	Sign-extension of byte data
SXH	reg1	I	–	–	–	–	–	Sign-extension of halfword data
SYNCE	(None)	I	–	–	–	–	–	Exception synchronize instruction
SYNCM	(None)	I	–	–	–	–	–	Memory synchronize instruction
SYNCP	(None)	I	–	–	–	–	–	Pipeline synchronize instruction
SYSCALL	vector8	X	–	–	–	–	–	System call exception
TRAP	vector5	X	–	–	–	–	–	Software exception
TST	reg1, reg2	I	–	0	0/1	0/1	–	Test
TST1	bit#3, disp16 [reg1]	VIII	–	–	–	0/1	–	Bit test
TST1	reg2, [reg1]	IX	–	–	–	0/1	–	Bit test
XOR	reg1, reg2	I	–	0	0/1	0/1	–	Exclusive OR
XORI	imm16, reg1, reg2	VI	–	0	0/1	0/1	–	Exclusive OR immediate
ZXB	reg1	I	–	–	–	–	–	Zero-extension of byte data
ZXH	reg1	I	–	–	–	–	–	Zero-extension of halfword data