

<Arithmetic instruction>

Arithmetic Instruction	
ADD	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]

(1) rrrrrr001110RRRRR

(2) rrrrrr010010iiii

[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]	<p>(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.</p> <p>(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.</p>
---------------	---

<Branch instruction>

JR	Jump relative
	Unconditional branch (PC relative)

[Instruction format] (1) JR disp22

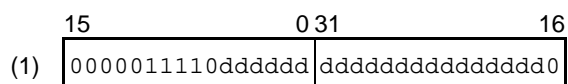
(2) JR disp32

[Operation] (1) $PC \leftarrow PC + \text{sign-extend}(\text{disp22})$ (2) $PC \leftarrow PC + \text{disp32}$

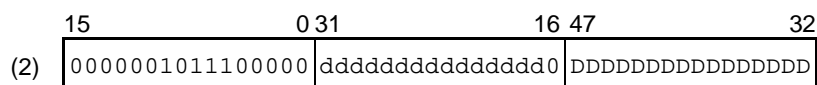
[Format] (1) Format V

(2) Format VI

[Opcode]



dddddddddddddddddd is the higher 21 bits of disp22.



DDDDDDDDDDDDDDDD is the higher 31 bits of disp32.

[Flags]

CY --

OV --

S --

Z --

SAT --

[Description]

(1) 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".

[Comment]

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.

<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)

15	0
rrrrr00000RRRRR	

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

(2)

15	0
rrrrr010000iiii	

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

(3)

15	0 31	16 47	32
00000110001RRRRR	iiiiiiiiiiiiiiii	IIIIIIIIIIIIIIII	

 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).

<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
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	31
32	32
33	33
34	34
35	35
36	36
37	37
38	38
39	39
40	40
41	41
42	42
43	43
44	44
45	45
46	46
47	47
48	48
49	49
50	50
51	51
52	52
53	53
54	54
55	55
56	56
57	57
58	58
59	59
60	60
61	61
62	62
63	63
64	64
65	65
66	66
67	67
68	68
69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100

[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.

<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>

Arithmetic Instructions	
SUBR	Subtract reverse
	Reverse subtraction

[Instruction format] SUBR reg1, reg2

[Operation]	GR [reg2] \leftarrow GR [reg1] – GR [reg2]
-------------	--

[Format]	Format I
----------	----------

[Opcode] 15 0
rrrrrr001100RRRRR

[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