<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
```

`dddddddd` 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]

```
      15              0
(1)  rrrrr001111RRRRR
      15              0
(2)  rrrrr010011iiiii
```

[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>

JR

Jump relative

Unconditional branch (PC relative)

[Instruction format] (1) JR disp22
(2) JR disp32

[Operation] (1) PC ← PC + sign-extend (disp22)
(2) PC ← PC + disp32

[Format] (1) Format V
(2) Format VI

[Opcode]

(1)

15 0	31 16
0000011110dddddd	ddddddddddddddd0

dddddddddddddddddddddd is the higher 21 bits of disp22.

(2)

15 0	31 16	47 32
0000001011100000	ddddddddddddddd0	DDDDDDDDDDDDDDDD

DDDDDDDDDDDDDDDDddddddddddddddd 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]

```
     15             0
(1) |rrrrr000000RRRRR|
```

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

```
     15             0
(2) |rrrrr010000iiiii|
```

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

```
     15             0 31            16 47            32
(3) |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).

<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`
(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.

<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.

<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]

```
     15              031              16
(1)  rrrrr111011RRRRR|ddddddddddddddd1
```

Where ddddddddddddddd is the higher 15 bits of disp16.

```
     15              031              1647             32
(2)  00000111100RRRRR|wwwwwdddddd01111|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 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.

<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]

```
     15              031              16
(1)  |rrrrr111001RRRRR|ddddddddddddddd1|
```

Where ddddddddddddddd is the higher 15 bits of disp16.

```
     15              031              1647             32
(2)  |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.

<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
0000000011RRRRR

(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.