MITSUBISHI 32-BIT SINGLE-CHIP MICROCOMPUTER

Mitsubishi Electric Corporation puts the maximum effort into making semiconductor products better and more reliable, but there is always the possibility that trouble may occur with them. Trouble with semiconductors may lead to personal injury, fire or property damage. Remember to give due consideration to safety when making your

CPU PROGRAMMING MODEL

1.4 Accumulator

The accumulator (ACC) is a 64-bit register used for the DSP function.
Use the **MVTACHI** and **MVTACLO**

1.6 Data format

1.6.2 Data formats

(1) Data format in a register

2.1 Instruction set overview

2.1.2 Transfer instructions

The transfer instructions carry out data transfers between registers or a register and an immediate value.

LD24	Load 24-bit immediate
LDI	Load immediate
MV	Move register
MVFC	

2.1 Instruction set overview

2.1.4 Branch instructions

The branch instructions are used to change the program flow.

BC	Branch on C-bit
BEQ	Branch on equal
BEQZ	Branch on equal zero
BGEZ	Branch on greater than or equal zero
BGTZ	Branch on greater than zero
BL	Branch and link
BLEZ	Branch on less than or equal zero
BLTZ	Branch on less than zero
BNC	Branch on not C-bit
BNE	Branch on not equal
BNEZ	Branch on not equal zero
BRA	Branch
JL	Jump and link
JMP	Jump
NOP	No operation

Only a word-aligned (word boundary) address can be specified for the branch address.

The addressing mode of the **BRA**, **BL**, **BC** and **BNC** instructions can specify an 8-bit or 24-bit immediate value. The addressing mode of the **BEQ**, **BNE**, **BEQZ**, **BNEZ**, **BLTZ**, **BGEZ**, **BLEZ**, and **BGTZ** instructions can specify a 16-bit immediate value.

In the **JMP** and **JL** instructions, the register value becomes the branch address. However, the low-order 2-bit value of the register is ignored. In other branch instructions, (PC value of branch instruction) + (sign-extended and 2 bits left-shifted immediate value) becomes the branch address. However, the low order 2-bit value of the address becomes "00" when addition is carried out. For example, refer to Figure 2.1.1. When instruction A or B is a branch instruction, branching to instruction G, the immediate value of either instruction A or B becomes 4.

Simultaneous with execution of branching by the **JL** or **BL**BL

2.1.5 EIT-related instructions

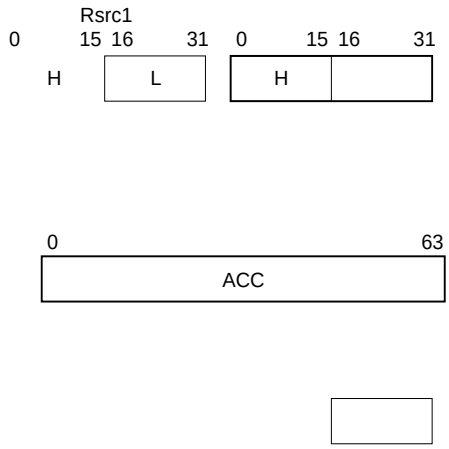


Fig. 2.1.2 DSP function instruction operation 1 (multiply, multiply and accumulate)

2.2 Instruction format

There are two major instruction formats: two 16-bit instructions packed together within a word boundary, and a single 32-bit instruction (see Fig. 2.2.1). Figure 2.2.2 shows the instruction format of M32R family.

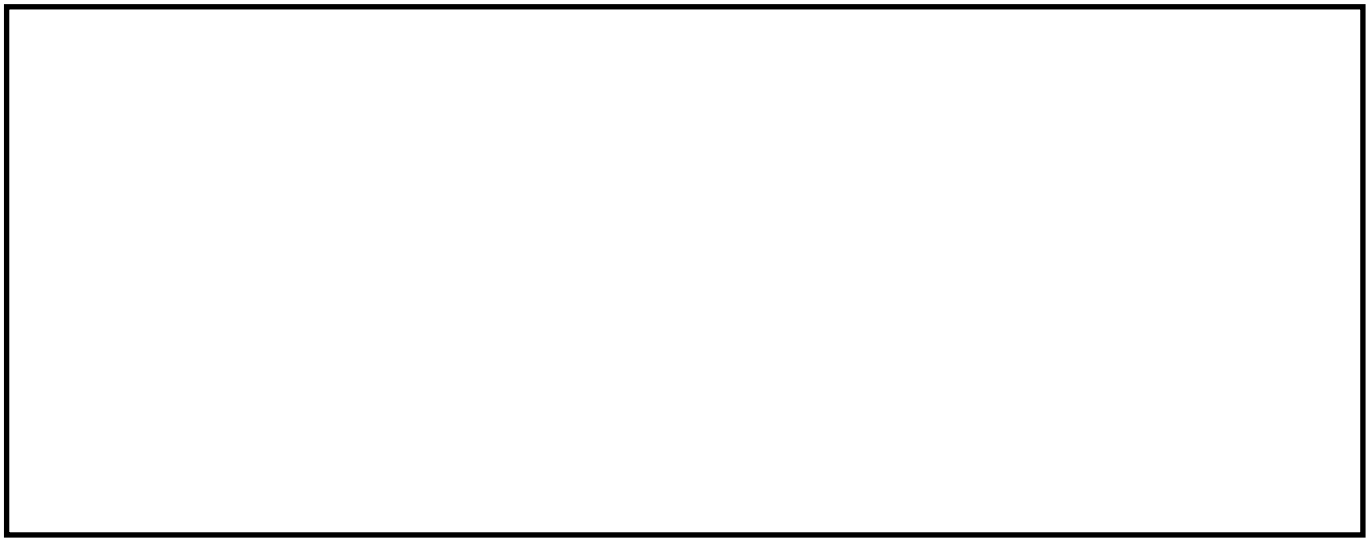


Table 3.1.3 Operation expression (operator) (cont.)

ADDI

arithmetic operation instruction

Add immediate

ADDI

[Mnemonic]

ADDI Rdest,#imm8

[Function]

Add

Rdest = Rdest + (signed char) imm8;

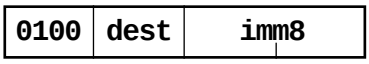
[Description]

ADDI adds the 8-bit immediate value to Rdest and puts the result in Rdest. The immediate value is sign-extended to 32 bits before the operation. The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]



ADDI Rdest,#imm8

ADDV

arithmetic operation instruction
Add with overflow checking

ADDV

[Mnemonic]

ADDV Rdest,Rsrc

[Function]

Add

```
Rdest = ( signed ) Rdest + ( signed ) Rsrc;
C = overflow  ?  1 : 0;
```

[Description]

ADDV adds Rsrc to Rdest and puts the result in Rdest.

INSTRUCTIONS

ADDX

BEQZ

branch instruction
Branch on equal zero

BEQZ

[Mnemonic]

```
BEQZ   Rsrc,pcdisp16
```

[Function]

BL

branch instruction

Branch and link

BL

BLEZ

branch instruction

Branch on less than or equal zero

BLEZ

[Mnemonic]

BLEZ Rsrc,pcdisp16

[Function]

Branch
if ((signed) Rsrc <= 0) PC = (PC & 0xfffffffc) + (((signed short) pcdisp16) << 2);

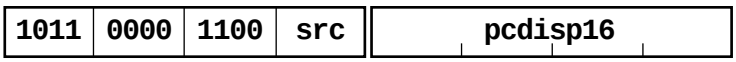
[Description]

BLEZ causes a branch to the specified label when the contents of Rsrc treated as a signed 32-bit value, is less than or equal to zero.
The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]



BLEZ Rsrc,pcdisp16

BLTZ

branch instruction
Branch on less than zero

BLTZ

[Mnemonic]

BLTZ Rsrc,pcdisp16

[Function]

Branch
if ((signed) Rsrc < 0) PC = (PC & 0xfffffffc) + (((signed short) pcdisp16) << 2);

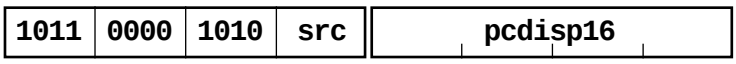
[Description]

BLTZ causes a branch to the specified label when Rsrc treated as a signed 32-bit value is less than zero.
The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]



BLTZ Rsrc,pcdisp16

CMP

compare instruction
Compare

CMP

[Mnemonic]

```
CMP   Rsrc1,Rsrc2
```

CMPI

compare instruction
Compare immediate

CMPI

[Mnemonic]

CMPI Rsrc,#imm16

[Function]

Compare
C = ((signed) Rsrc < (signed short) imm16) ? 1:0;

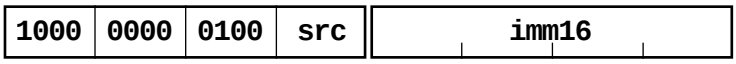
[Description]

The condition bit (C) is set when Rsrc is less than 16-bit immediate value. The operands are treated as signed 32-bit values. The immediate value is sign-extended to 32-bit before the operation.

[EIT occurrence]

None

[Encoding]



CMPI Rsrc,#imm16

JMP

branch instruction
Jump

JMP

[Mnemonic]

INSTRUCTIONS

LD24

load/store instruction
Load 24-bit immediate

LD24

[Mnemonic]

LD24 Rdest,#imm24

[Function]

Load
Rdest = imm24 & 0x00ffffff;

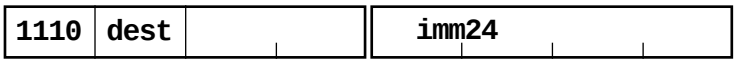
[Description]

LD24 loads the 24-bit immediate value into Rdest. The immediate value is zero-extended to 32 bits.
The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]



LD24 Rdest,#imm24

load/store instruction

Load byte

LDB

[Mnemonic]

① `LDB Rdest,@Rsrc`
② `LDB Rdest,@(disp16,Rsrc)`

[Function]

MULHI

DSP function instruction

MULHI

[Mnemonic]

```
MULHI   Rsrc1,Rsrc2
```

[Function]

Multiply

accumulator = ((signed) (Rsrc1 & 0xffff000) * (signed short) (Rsrc2 >> 16));

[Description]

MULHI multiplies the high-order 16 bits of Rsrc1 and the high-order 16 bits of Rsrc2, and stores the result in the accumulator.
However, the LSB of the multiplication result is aligned with bit 47 in the accumulator, and the portion corresponding to bits 0 through 15 of the accumulator is sign-extended. Bits 48 through 63 of the accumulator are cleared to 0. The high-order 16 bits of Rsrc1 and Rsrc2 are treated as signed values.
The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]

[Mnemonic]

```
MULLO   Rsrc1,Rsrc2
```

MVFACHI

DSP function instruction
Move from accumulator

MVFACHI

MVFACMI *DSP function instruction* MVFACMI

Move from accumulator middle-order word

[Mnemonic]

MVFACMI Rdest

[Function]

Transfer from accumulator to register

Rdest = (int) (accumulator >> 16) ;

[Description]

MVFACMI moves bits16 through 47 of the accumulator to Rdest. The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]

0101	**dest**		**0010**

MVTC

transfer instruction
Move to control register

MVTC

[Mnemonic]

```
MVTC   Rsrc,CRdest
```

[Function]

Transfer from register to control register
CRdest = Rsrc ;

[Description]

MVTC moves Rsrc to CRdest.
If PSW(CR0) is specified as CRdest, the condition bit (C) is changed; otherwise it is unchanged.

[EIT occurrence]

None

[Encoding]

0001	dest	1010	src

MVTC Rsrc,CRdest

NEG

arithmetic operation instruction
Negate

NEG

[Mnemonic]

```
NEG   Rdest,Rsrc
```

[Function]

NOP

branch instruction
No operation

NOP

[Mnemonic]

NOP

[Function]

No operation
/* */

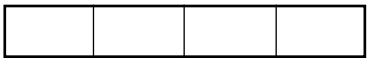
[Description]

NOP performs no operation. The subsequent instruction then processed. The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]



NOT

logic operation instruction
Logical NOT

NOT

[Mnemonic]

```
NOT   Rdest,Rsrc
```

[Function]

Logical NOT
Rdest = ~ Rsrc ;

[Description]

NOT inverts each of the bits of Rsrc and puts the result in Rdest. The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]

0000	dest	1011	src

NOT Rdest,Rsrc

[Supplement]

This instruction is executed in two steps as shown below:

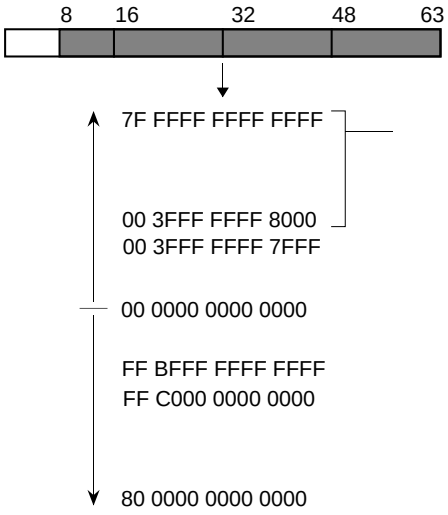
<step 1>

<step 2>

[Supplement]

This instruction is executed in two steps, as shown below.

<step 1>



<step 2>

0

REM

multiply and divide instruction
Remainder

REM

[Mnemonic]

REM Rdest,Rsrc

[Function]

Signed division
Rdest = (signed) Rdest % (signed) Rsrc ;

[Description]

REM divides Rdest by Rsrc and puts the quotient in Rdest. The operands are treated as signed 32-bit values.
The quotient is rounded toward zero and the quotient takes the same sign as the dividend.
The condition bit (C) is unchanged.
When Rsrc is zero, Rdest is unchanged.

[EIT occurrence]

None

[Encoding]

1001	dest	0010	src	0000	0000	0000	0000

REM Rdest,Rsrc

INSTRUCTIONS

SLL

shift instruction

Shift left logical

SLL

SLL3

shift instruction

Shift left logical 3-operand

SLL3

[Mnemonic]

SLL3 Rdest,Rsrc,#imm16

[Function]

Logical left shift

```
Rdest = Rsrc << ( imm16 & 31 ) ;
```

[Description]

SLL3 left logical-shifts the contents of Rsrc into Rdest by the number specified by the 16-bit immediate value, shifting zeroes into the least significant bits.
Only the five least significant bits of the 16-bit immediate value are used.
The condition bit (C) is unchanged.

[EIT occurrence]

None

[Encoding]

SRA3

shift instruction

Shift right arithmetic 3-operand

SRA3

[Mnemonic]

SRLI

shift instruction

Shift right logical immediate

SRLI

[Mnemonic]

```
SRLI   Rdest,#imm5
```

[Function]

Logical right shift

```
Rdest = ( unsigned ) Rdest  >>  ( imm5 & 31 ) ;
```

[Description]

SRLI right arithmetic-shifts Rdest by the number specified by the 5-bit immediate value, shifting zeroes into the most significant bits.
The condition bit (C) is unchanged.

[EIT occurrence]

None

SUB

arithmetic operation instruction
Subtract

SUB

[Mnemonic]

```
SUB   Rdest,Rsrc
```

[Function]

Subtract

SUBX

arithmetic operation instruction

SUBX

TRAP

EIT-related instruction
Trap

TRAP

A

Appendix A Instruction list

B.2 Instructions and pipeline processing

The M32R pipeline has five stages. However, the MEM stage is used only when the load/store instruction is eqstruction

APPENDICES

Rev.

MITSUBISHI 32-BIT SINGLE-CHIP MICROCOMPUTER
M32R Family Software Manual

July 1998 : Revised edition

Printed in Japan

Revised publication, effective July 1998