

4 MAC INSTRUCTIONS

The instruction set provides MAC instructions for performing high-speed multiplication and multiply with cumulative add/subtract operations. MAC instructions include:

- “Multiply” on page 4-8
- “Multiply with Cumulative Add” on page 4-11
- “Multiply with Cumulative Subtract” on page 4-14
- “MAC Clear” on page 4-17
- “MAC Round/Transfer” on page 4-19
- “MAC Saturate” on page 4-21
- “Generate MAC Status Only: NONE” on page 4-24

This chapter describes the individual MAC instructions and the following related topics:

- “MAC Input Registers” on page 4-2
- “MAC Output Registers” on page 4-2
- “Data Format Options” on page 4-3
- “Status Flags” on page 4-7

For details on condition codes and data input and output registers, see “Condition Codes” on page 9-11 and “Core Register Codes” on page 9-13.

MAC Input Registers

All unconditional, single-function multiply and multiply with accumulative add or subtract instructions can use any DREG data register for the x and y input operands (for details, see “Core Register Codes” on page 9-13). So, the program can use, for example, the ALU registers for the multiplication or shifter operations, without issuing a separate data move instruction. This capability simplifies register allocation in algorithm coding. For example, using the DSP’s dual accumulator:

$$SR = SR + MX0 * MY0 (SS);$$

But in multifunction operations, you can use only certain registers for the x -input operand (AR, MX0, MX1, MR0, MR1, MR2, SR0, SR1) and the y -input operand (MY0, MY1, SR1, 0).

All conditional MAC instructions must use the restricted x_{op} and y_{op} data registers for the x and y input operands, or an x_{op} register for the x -input and 0 for the y -input.

MAC Output Registers

All MAC instructions can use the multiplier MR output registers or the shifter SR output registers to receive the result of a multiplier operation. Availability of the shifter SR output registers for multiplier operations provides dual accumulator functionality.

When MR is the result register, results are directly available from MR0, MR1, or MR2 as the x -input operand into the very next multiplier operation.

$$MR = MR + AX0 * AX0 (SS);$$

When SR is the result register, the 16-bit value in SR1 (bits 31:16 of the 40-bit result) is directly available as the y -input operand into the very next multiplier operation. This functionality is most useful when shifting the

results of a multiply/accumulate operation since it decreases the number of required data moves.

$$SR = SR + AX0 * AYO (SS);$$

$$SR = SR + SR1 * AYO (SS);$$

Data Format Options

Multiplier operations require the instruction to specify the data format of the input operands (either signed or unsigned) or specify that the multiplier rounds (RND) the product of two signed operands.

All data format options, except the round option (RND), which affects the product stored in the result register, specify the format of both input operands in x/y order. The data format options are:

- (RND) Round value in result register.

When overflow occurs, rounds the product to the most significant twenty-four bits—SR2/SR1 or MR2/MR1 represent the rounded 24-bit result. Otherwise, rounds bits 31:16 to sixteen bits—MR1 or SR1 contain the rounded 16-bit result.

With (RND) selected, the multiplier considers both input operands signed (twos complement). If the DSP is in fractional mode (MSTAT:M_MODE = 0), the multiplier rounds the result after adjusting for fractional data format. For details, see [“Numeric Format Modes” on page 4-6](#).

The DSP provides two rounding modes, biased and unbiased, to support a variety of application algorithms. For details, see [“Rounding Modes” on page 4-4](#).

MAC Instructions

- (SS) Both input operands are signed numbers. Signed numbers are in twos complement format.

You use this option to multiply two signed single-precision numbers or to multiply the upper portions of two signed multiprecision numbers.

- (SU) X-input operand is signed; y-input operand is unsigned.

You use this option to multiply a signed single-precision number by an unsigned single-precision number.

- (US) X-input operand is unsigned; y-input operand is signed.

You use this option to multiply an unsigned single-precision number by a signed single-precision number.

- (UU) Both input operands are unsigned numbers. Unsigned numbers are in ones complement format.

You use this option to multiply two unsigned single-precision numbers or to multiply the lower portions of two signed multiprecision numbers.

Rounding Modes

Rounding operates on the boundary between bits 15 and 16 of the 40-bit adder result. The multiplier directs the rounded output to either the MR or the SR result registers.

The ADSP-219x provides two modes for rounding. The rounding algorithm is the same for both modes, but the final results can differ when the product equals the midway value ($MR0 = 0 \times 8000$).

In both methods, the multiplier adds 1 to value of bit 15 in the adder chain. But when $MR0 = 0 \times 8000$, the multiplier forces bit 16 in the result output to 0. Although applied on every rounding operation, the result of this algorithm is evident only when $MR0 = 0 \times 8000$ in the adder chain.

The rounding mode determines the final result. The `BIASRND` bit in the `ICNTL` register selects the mode. `BIASRND = 0` selects unbiased rounding, and `BIASRND = 1` selects biased rounding.

- **Unbiased rounding** Default mode. Rounds up only when `MR1/SR1` set to an odd value; otherwise, rounds down. Yields a zero large-sample bias.
- **Biased rounding** Always rounds up when `MR0/SR0` is set to `0x8000`.

Table 4-1 shows the results of rounding for both modes.

Table 4-1. MR result values

| MR Value before RND | Biased RND Result | Unbiased RND Result |
|---------------------|-------------------|---------------------|
| 00-0000-8000 | 00-0001-0000 | 00-0000-0000 |
| 00-0001-8000 | 00-0002-0000 | 00-0002-0000 |
| 00-0000-8001 | 00-0001-0001 | 00-0001-0001 |
| 00-0001-8001 | 00-0002-0001 | 00-0002-0001 |
| 00-0000-7FFF | 00-0000-FFFF | 00-0000-FFFF |
| 00-0001-7FFF | 00-0001-FFFF | 00-0001-FFFF |

Unbiased rounding, preferred for most algorithms, yields a zero large-sample bias, assuming uniformly distributed values. Biased rounding supports efficient implementation of bit-specified algorithms, such as GSM speech compression routines.

MAC Instructions

Numeric Format Modes

The multiplier can operate on integers or fractions. The `M_MODE` bit in the `MSTAT` register selects the mode. `M_MODE = 0` selects fractional mode, and `M_MODE = 1` selects integer mode.

The mode determines whether the multiplier shifts the product before adding or subtracting it from the result register.

- Integer mode 16.0 integer format

The LSB of the 32-bit product is aligned with the LSB of `MR0/SR0`.

In multiply and accumulate operations, the multiplier sign-extends the 32-bit product (8 bits) then adds or subtracts that value from the result register to form the new 40-bit result.

The multiplier sets the `MV/SV` overflow bit when the result falls outside the range of -1 to $+1-2^{31}$.

- Fractional mode 1.15 fraction format

Fractions range from -1 to $+1-2^{15}$. The MSB of the product is aligned with the MSB of `MR1/SR1`. `MR1-0/SR1-0` hold a 32-bit fraction (1.31 format) in the range of -1 to $+1-2^{31}$, while `MR2/SR2` contains the eight sign-extended bits. In total, the `MR/SR` registers contains a fraction in 9.31 format.

In multiply and accumulate operations, the multiplier adjusts the format of the 32-bit product before adding or subtracting it from the result register. To do so, the multiplier sign-extends the product (7 bits), shifts it one bit to the left, and then adds or subtracts that value from the result register to form the new 40-bit result.

The multiplier sets the `MV/SV` overflow bit when the result falls outside the range of -1 to $+1-2^{31}$.

Status Flags

Two status flags in the `ASTAT` register record the status of multiplier operations. `MV = 1` records an overflow or underflow state when `MR` is the specified result register, and `SV = 1` records an overflow or underflow state when `SR` is the specified result register.

MAC Instructions

Multiply

$$\left| \begin{array}{l} \text{MR} \\ \text{SR} \end{array} \right| = \text{DREG1} * \text{DREG2} \left(\left| \begin{array}{l} \text{RND} \\ \text{SS} \\ \text{SU} \\ \text{US} \\ \text{UU} \end{array} \right| \right) ;$$
$$[\text{IF COND}] \left| \begin{array}{l} \text{MR} \\ \text{SR} \end{array} \right| = \text{XOP} * \text{YOP} \left(\left| \begin{array}{l} \text{RND} \\ \text{SS} \\ \text{SU} \\ \text{US} \\ \text{UU} \end{array} \right| \right) ;$$

FUNCTION

Multiplies the input operands and stores the result in the specified result register. Optionally, inputs may be signed or unsigned, and output may be rounded. For more information on input and output options, see [“Data Format Options” on page 4-3](#).

If execution is based on a condition, the multiplier performs the multiplication only if the condition evaluates true, and it performs a NOP operation if the condition evaluates false. Omitting the condition forces unconditional execution of the instruction.

INPUT

For the unconditional form of this instruction, you can use any of these data registers for the DREG inputs:

| |
|--|
| Register File |
| AX0, AX1, AY0, AY1, AR, MX0, MX1, MY0, MY1, MR0, MR1, MR2, SR0, SR1, SR2, SI |

For the conditional form of this instruction, the input operands are restricted. Valid XOP and YOP registers are:

| | |
|---------------------------------------|------------------|
| Xops | Yops |
| AR, MX0, MX1, MR0, MR1, MR2, SR0, SR1 | MY0, MY1, SR1, 0 |

OUTPUT

- MR Multiplier result register. Results are directly available for x input only in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.
- SR Multiplier feedback register. Results are directly available for either x (SR0 and SR1) or y (SR1 only) input in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.

STATUS FLAGS

| | |
|---|----------------------------|
| Affected Flags—set or cleared by the operation | Unaffected Flags |
| MV (if MR used), SV (if SR used) | AZ, AN, AV, AC, AS, AQ, SS |
| For information on these status bits in the ASTAT register, see Table 2-2 on page 2-5 . | |

MAC Instructions

DETAILS

This instruction provides a squaring operation ($XOP * XOP$ and $DREG1 * DREG1$) that performs single-cycle X^2 and ΣX^2 functions. In squaring operations, you must use the same register for both x-input operands.

You cannot use DREG form of the multiply instruction in multifunction instructions.

EXAMPLES

```
MR = AYO * SI (RND);           /* mult DREGs, round result */
SR = AXO * MX1 (SS);           /* mult DREGs, signed inputs */

IF MV MR = MXO * MYO (SU);     /* mult signed X, unsigned Y */
CCODE = 0x09; NOP;             /* set CCODE for SV condition */
IF SWCOND SR = MRO * SR1 (UU); /* mult unsigned X and Y */
```

SEE ALSO

- [“Type 9: Compute” on page 9-27](#)
- [“Condition Code \(CCODE\) Register” on page 2-6](#)
- [“Mode Status \(MSTAT\) Register” on page 2-11](#)

Multiply with Cumulative Add

| | | | | | | | | | | |
|----|---|----|---|-------|---|-------|---|-----|---|---|
| MR | = | MR | + | DREG1 | * | DREG2 | (| RND |) | ; |
| SR | = | SR | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |

| | | | | | | | | | | | |
|-----------|----|---|----|---|-----|---|-----|---|-----|---|---|
| [IF COND] | MR | = | MR | + | XOP | * | YOP | (| RND |) | ; |
| | SR | = | SR | | YOP | | XOP | | SS | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |
| | | | | | | | | | | | |

FUNCTION

Multiplies the input operands, adds the product to the current contents of the MR or SR register, and then stores the sum in the corresponding result register. Optionally, inputs may be signed or unsigned, and output may be rounded. For more information on input and output options, see [“Data Format Options” on page 4-3](#).

If execution is based on a condition, the multiplier performs the operation only if the condition evaluates true, and it performs a NOP operation if the condition evaluates false. Omitting the condition forces unconditional execution of the instruction.

MAC Instructions

INPUT

For the unconditional form of this instruction, you can use any of these data registers for the DREG inputs:

| Register File |
|--|
| AX0, AX1, AY0, AY1, AR, MX0, MX1, MY0, MY1, MR0, MR1, MR2, SR0, SR1, SR2, SI |

For the conditional form of this instruction, the input operands are restricted. Valid XOP and YOP registers are:

| Xops | Yops |
|---------------------------------------|------------------|
| AR, MX0, MX1, MR0, MR1, MR2, SR0, SR1 | MY0, MY1, SR1, 0 |

OUTPUT

- MR Multiplier result register. Results are directly available for x input only in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.
- SR Multiplier feedback register. Results are directly available for either x (SR0 and SR1) or y (SR1 only) input in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.

STATUS FLAGS

| Affected Flags—set or cleared by the operation | Unaffected Flags |
|---|----------------------------|
| MV (if MR used), SV (if SR used) | AZ, AN, AV, AC, AS, AQ, SS |
| For information on these status bits in the ASTAT register, see Table 2-2 on page 2-5 . | |

DETAILS

This instruction provides a squaring operation ($x_{op} * x_{op}$ and $DREG * DREG$) that performs single-cycle X^2 and ΣX^2 functions. In squaring operations, you must use the same register for both x-input operands.

You cannot use unconditional ($Dreg$) form of the multiply instruction in multifunction instructions.

EXAMPLES

```
MR = MR + AX0 * SI (RND); /* mult DREGs, rnd output, sum */
SR = SR + AX1 * MX0 (SS); /* mult DREGs, sign input, sum */

IF MV MR = MR + MR0 * MY0 (SU);
                               /* mult X/Yops, un/sign in, sum */
IF MV MR = MR + MR2 * MX1 (UU);
                               /* mult X/Yops, un/sign in, sum */
CCODE = 0x09; NOP;             /* set CCODE for SV condition */
IF SWCOND SR=SR+SR0*MY0 (US);
                               /* mult X/Yops, un/sign in, sum */
```

SEE ALSO

- [“Type 9: Compute” on page 9-27](#)
- [“Condition Code \(CCODE\) Register” on page 2-6](#)
- [“Mode Status \(MSTAT\) Register” on page 2-11](#)

Multiply with Cumulative Subtract

| | | | | | | | | | | | | | | |
|--|---|----|---|-------|---|-------|---|-----|---|---|----|----|----|----|
| MR | = | MR | - | DREG1 | * | DREG2 | (| RND |) | ; | | | | |
| SR | = | SR | | | | | | | | | | | | |
| <table style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr><td style="border-right: 1px solid black; padding: 0 5px;">SS</td></tr> <tr><td style="border-right: 1px solid black; padding: 0 5px;">SU</td></tr> <tr><td style="border-right: 1px solid black; padding: 0 5px;">US</td></tr> <tr><td style="border-right: 1px solid black; padding: 0 5px;">UU</td></tr> </table> | | | | | | | | | | | SS | SU | US | UU |
| SS | | | | | | | | | | | | | | |
| SU | | | | | | | | | | | | | | |
| US | | | | | | | | | | | | | | |
| UU | | | | | | | | | | | | | | |

| | | | | | | | | | | | | | |
|--|---|----|---|----|---|-----|---|-----|---|-----|----|----|----|
| [IF COND] | (| MR | = | MR | - | XOP | * | YOP | (| RND |) | ; | |
| | | SR | = | SR | - | YOP | * | XOP | (| SS |) | | |
| <table style="border-collapse: collapse; margin-left: auto; margin-right: auto;"> <tr><td style="border-right: 1px solid black; padding: 0 5px;">SU</td></tr> <tr><td style="border-right: 1px solid black; padding: 0 5px;">US</td></tr> <tr><td style="border-right: 1px solid black; padding: 0 5px;">UU</td></tr> </table> | | | | | | | | | | | SU | US | UU |
| SU | | | | | | | | | | | | | |
| US | | | | | | | | | | | | | |
| UU | | | | | | | | | | | | | |

FUNCTION

Multiplies the input operands, subtracts the product from the current contents of the MR or SR register, and then stores the result in the corresponding destination register. Optionally, inputs may be signed or unsigned, and output may be rounded. For more information on input and output options, see [“Data Format Options” on page 4-3](#).

If execution is based on a condition, the multiplier performs the operation only if the condition evaluates true, and it performs a NOP operation if the condition evaluates false. Omitting the condition forces unconditional execution of the instruction.

Multiply with Cumulative Subtract

INPUT

For the unconditional form of this instruction, you can use any of these data registers for the DREG inputs:

| Register File |
|--|
| AX0, AX1, AY0, AY1, AR, MX0, MX1, MY0, MY1, MR0, MR1, MR2, SR0, SR1, SR2, SI |

For the conditional form of this instruction, the input operands are restricted. Valid XOP and YOP registers are:

| Xops | Yops |
|---------------------------------------|------------------|
| AR, MX0, MX1, MR0, MR1, MR2, SR0, SR1 | MY0, MY1, SR1, 0 |

OUTPUT

- MR Multiplier result register. Results are directly available for x input only in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.
- SR Multiplier feedback register. Results are directly available for either x (SR0 and SR1) or y (SR1 only) input in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.

STATUS FLAGS

| Affected Flags—set or cleared by the operation | Unaffected Flags |
|---|----------------------------|
| MV (if MR used), SV (if SR used) | AZ, AN, AV, AC, AS, AQ, SS |
| For information on these status bits in the ASTAT register, see Table 2-2 on page 2-5 . | |

MAC Instructions

DETAILS

This instruction provides a squaring operation ($x_{op} * x_{op}$ and $DREG * DREG$) that performs single-cycle X^2 and ΣX^2 functions. In squaring operations, you must use the same register for both x input operands.

You cannot use unconditional ($Dreg$) form of the multiply instruction in multifunction instructions.

EXAMPLES

```
MR = MR - AX0 * SI (RND); /* mult DREGs, rnd output, sub */
SR = SR - AX1 * MX0 (SS); /* mult DREGs, sign input, sub */

IF MV MR = MR - MR0 * MY0 (SU);
                               /* mult X/Yops, un/sign in, sub */
IF MV MR = MR - MR2 * MX1 (UU);
                               /* mult X/Yops, un/sign in, sub */
CCODE = 0x09; NOP;           /* set CCODE for SV condition */
IF SWCOND SR=SR-SR0*MY0 (US);
                               /* mult X/Yops, un/sign in, sub */
```

SEE ALSO

- [“Type 9: Compute” on page 9-27](#)
- [“Condition Code \(CCODE\) Register” on page 2-6](#)
- [“Mode Status \(MSTAT\) Register” on page 2-11](#)

MAC Clear

| | | |
|-------------|----|-------|
| [IF COND] | MR | = 0 ; |
| | SR | |

FUNCTION

Sets the specified register to 0.

If execution is based on a condition, the multiplier performs the operation only if the condition evaluates true, and it performs a NOP operation if the condition evaluates false. Omitting the condition forces unconditional execution of the instruction.

INPUT

This instruction is a special case of $x_{op} * y_{op}$ with the y -input operand set to 0.

OUTPUT

- MR Multiplier result register. Results are directly available for x input only in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.
- SR Multiplier feedback register. Results are directly available for either x (SR0 and SR1) or y (SR1 only) input in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.

MAC Instructions

STATUS FLAGS

| Affected Flags—set or cleared by the operation | Unaffected Flags |
|---|----------------------------|
| MV (cleared if MR used), SV (cleared if SR used) | AZ, AN, AV, AC, AS, AQ, SS |
| For information on these status bits in the ASTAT register, see Table 2-2 on page 2-5 . | |

DETAILS

See description in [“function” on page 4-17](#).

EXAMPLES

```
MR = 0;          /* clears MR */
SR = 0;          /* clears SR */
```

SEE ALSO

- [“Type 9: Compute” on page 9-27](#)
- [“Condition Code \(CCODE\) Register” on page 2-6](#)
- [“Mode Status \(MSTAT\) Register” on page 2-11](#)

MAC Round/Transfer

[IF COND] MR = MR (RND) ;

[IF COND] SR = SR (RND) ;

FUNCTION

Performs a multiply with cumulative add operation in which the y -input operand is 0 and the zero-product is added to the specified result register. Rounding (RND) directs the multiplier to round the entire 40-bit value stored in the result register, MR or SR.

If execution is based on a condition, the multiplier performs the operation only if the condition evaluates true, and it performs a NOP operation if the condition evaluates false. Omitting the condition forces unconditional execution of the instruction.

INPUT

This instruction is a special case of $MR|SR + x_{op} * y_{op}$ with the y -input operand set to 0.

OUTPUT

- MR Multiplier result register. Results are directly available for x input only in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.
- SR Multiplier feedback register. Results are directly available for either x (SR0 and SR1) or y (SR1 only) input in the next conditional ALU, MAC, or shifter operation or as either x or y input in the next unconditional ALU, MAC, or shifter operation.

MAC Instructions

STATUS FLAGS

| Affected Flags—set or cleared by the operation | Unaffected Flags |
|---|----------------------------|
| MV (if MR used), SV (if SR used) | AZ, AN, AV, AC, AS, AQ, SS |
| For information on these status bits in the ASTAT register, see Table 2-2 on page 2-5 . | |

DETAILS

The `BIASRND` bit in the `ICNTL` register determines the rounding mode. Refer to the section [“Rounding Modes” on page 4-4](#) for more information. For a complete description of the MAC Round/ Transfer instruction, see the section [“function” on page 4-19](#).

EXAMPLES

```
MR = MR (RND);           /* round MR */
IF EQ SR = SR (RND);    /* round SR */
```

SEE ALSO

- [“Type 9: Compute” on page 9-27](#)
- [“Condition Code \(CCODE\) Register” on page 2-6](#)
- [“Mode Status \(MSTAT\) Register” on page 2-11](#)

MAC Saturate

SAT MR ;

SAT SR ;

FUNCTION

Tests the *MV* (MAC overflow) or *SV* (shifter overflow) bit in the *ASTAT* register. If set (1), the multiplier saturates the low-order (31:0) bits of the 40-bit *MR* or *SR* register; otherwise, the multiplier performs a *NOP* operation.

INPUT

None.

OUTPUT

MR Multiplier result register. Results are directly available for *x* input only in the next conditional ALU, MAC, or shifter operation or as either *x* or *y* input in the next unconditional ALU, MAC, or shifter operation.

SR Multiplier feedback register. Results are directly available for either *x* (*SR0* and *SR1*) or *y* (*SR1* only) input in the next conditional ALU, MAC, or shifter operation or as either *x* or *y* input in the next unconditional ALU, MAC, or shifter operation.

STATUS FLAGS

| Affected Flags—set or cleared by the operation | Unaffected Flags |
|--|------------------------------------|
| | AZ, AN, AV, AC, AS, AQ, SS, MV, SV |
| For information on these status bits in the <i>ASTAT</i> register, see Table 2-2 on page 2-5 . | |

MAC Instructions

DETAILS

The MAC saturation instruction provides control over a multiplication result that has overflowed or underflowed. It saturates the value in the specified register only for the cycle in which it executes. It does not enable a mode that continuously saturates results until disabled, like the ALU. Used at the end of a series of multiply and accumulate operations, the saturation instruction prevents the accumulator from overflowing.

For every operation it performs, the multiplier generates an overflow status signal MV (SV when SR is the specified result register), which is recorded in the ASTAT status register. $MV = 1$ when the accumulator result, interpreted as a signed (twos complement) number, crosses the 32-bit boundary, spilling over from MR1 into MR2. That is, the multiplier sets $MV = 1$ when the upper nine bits in MR are anything other than all 0s or all 1s. Otherwise, it sets $MV = 0$.

The operation invoked by the saturation instruction depends on the overflow status bit MV (or SV) and the MSB of MR2, which appear in Table 4-2. If $MV/SV = 0$, no saturation occurs. When $MV/SV = 1$, the multiplier examines the MSB of MR2 to determine whether the result has overflowed or underflowed. If the MSB = 0, the result has overflowed, and the multiplier saturates the MR register, setting it to the maximum positive value. If the MSB = 1, the result has underflowed, and the multiplier saturates the MR register, setting it to the maximum negative value.

Table 4-2. Saturation Status Bits & Result Registers

| MV/SV | MSB of MR2/SR2 | MR/SR Results |
|-------|----------------|--|
| 0 | 0 | No change. |
| 0 | 1 | No change. |
| 1 | 0 | 00000000 0111111111111111 1111111111111111 |
| 1 | 1 | 11111111 1000000000000000 0000000000000000 |

Do not permit the result to overflow beyond the MSB of $MR2$. Otherwise, the true sign bit of the result is irretrievably lost, and saturation may not produce a correct result. To reach this state, however, takes more than 255 overflows (MV type).

EXAMPLES

```
SAT MR;           /* saturate MR */  
SAT SR;           /* saturate SR */
```

SEE ALSO

- [“Type 9: Compute” on page 9-27](#)
- [“Condition Code \(CCODE\) Register” on page 2-6](#)
- [“Mode Status \(MSTAT\) Register” on page 2-11](#)

MAC Instructions

Generate MAC Status Only: NONE

[NONE =] <MAC Operation> ;

FUNCTION

Performs the indicated unconditional MAC operation but does not load the results into the MR or SR result registers. Generates MAC status flags only. You can use this instruction to set MAC status without disturbing the contents of the MR and SR result registers.

INPUT

XOP Limits the registers for the x input operand. Valid XOP registers are:

| |
|---------------------------------------|
| Xops |
| AR, MX0, MX1, MR0, MR1, MR2, SR0, SR1 |

YOP Limits the registers for the y input operand. Valid YOP registers are:

| |
|------------------|
| Yops |
| MY0, MY1, SR1, 0 |

OUTPUT

None. Generates MAC status flags only.

STATUS FLAGS

| Affected Flags—set or cleared by the operation | Unaffected Flags |
|---|--------------------------------|
| MV | AZ, AN, AV, AC, AS, AQ, SS, SV |
| For information on these status bits in the ASTAT register, see Table 2-2 on page 2-5 . | |

DETAILS

You can use any unconditional MAC operation to generate MAC status flags.

EXAMPLES

```
MX0 * MY0;                               /* generate status from mult */
```

SEE ALSO

- [“Type 8: Compute | Dreg1 «... Dreg2” on page 9-26.](#)

MAC Instructions