Chapter 12
SSE Floating-Point Instructions

The SSE floating-point instructions were introduced by Intel in 1998 and have continually expanded ever since. They operate on single-precision or double-precision data (Definition 5.3) residing in the 128-bit *XMM* registers or the 256-bit *YMM* registers. Some SSE instructions are *packed*, i.e., they partition their operands into several floating-point encodings to be processed in parallel; others are *scalar*, performing a single operation, usually on data residing in the low-order bits of their register arguments. The specifications presented in this chapter apply to both scalar and packed instructions that perform the operations of addition, multiplication, division, square root extraction, and FMA.

A single dedicated 16-bit register, the *MXCSR*, controls and records the response to exceptional conditions that arise during the execution of SSE floating-point operations and controls the rounding of floating-point values.

12.1 SSE Control and Status Register

The MXCSR bits are named as displayed in Fig. 12.1.

- The least significant six bits are the *exception flags*, corresponding to the pre-computation exceptions, invalid operand (IE), denormal operand (DE), and division by zero (ZE); and the post-computation exceptions, overflow (OE), underflow (UE), and inexact result (PE).
- Bit 6 is the *denormal-as-zero* bit, which, if set, coerces all denormal inputs to ± 0.
- Bits 12:7 are the *exception masks* corresponding to the flags, which determine whether an exceptional condition results in the return of a default value or the generation of an exception.

D. M. Russinoff, *Formal Verification of Floating-Point Hardware Design*,
https://doi.org/10.1007/978-3-319-95513-1_12

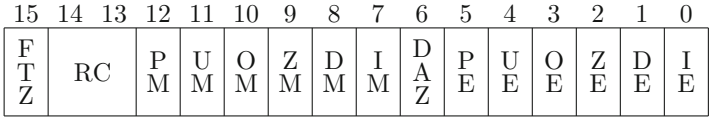

Fig. 12.1 MXCSR: SSE floating-point control and status register

Encoding	Rounding mode
00	*RNE*
01	*RDN*
10	*RUP*
11	*RTZ*

Table 12.1 x86 rounding control

- Bits 14:13 form the *rounding control* field, which encodes a rounding mode as displayed in Table 12.1.
- Bit 15 is the *force-to-zero* bit, which, if set, coerces any denormal output to ± 0.

12.2 Overview of SSE Floating-Point Exceptions

When an exceptional floating-point condition is detected during the execution of an SSE instruction, one of the exception flags MXCSR[5:0] may be set. If a flag is set and the exception is *unmasked*, i.e., the corresponding mask bit in MXCSR[12:7] is 0, then execution of the instruction is terminated, no value is written to the destination, and control is passed to a trap handling routine. If the exception is *masked*, then depending on the exceptional condition, either the instruction proceeds normally or a default value is returned, allowing execution of the program to proceed. For a packed instruction, if any of the component operations results in an unmasked exception, then no result is written for any operation. Otherwise, a result is written for each operation.

Instruction execution consists of three phases: pre-computation, computation, and post-computation. The exceptional conditions are partitioned into two classes, which are detected during the first and third of these phases. The following procedure is followed by both packed and scalar SSE floating-point instructions. Note that in the case of a packed instruction, the procedure is complicated by the requirement of parallel execution:

- Pre-Computation (Sect. 12.3): The operands of each operation are examined in parallel for a set of conditions, some of which result in the setting of a flag, IE, DE, or ZE. If a flag is set and the corresponding mask is clear, then execution is terminated for all operations and no value is written to the destination. Otherwise, for each operation, either a QNaN is selected as a default value (but not yet written), or the computation proceeds.

- Computation (Sect. 12.4): Unless an unmasked exception is detected during the pre-computation phase, for each operation that has not terminated, a computation is performed. If the value is infinite or 0, then a result is determined (but not yet written). Otherwise, execution proceeds.
- Post-Computation (Sect. 12.5): For each remaining operation, the computed value is rounded and the result is examined for a set of conditions, which may result in the setting of one or two of the flags OE, UE, and PE. If a flag is set and the corresponding mask bit is 1, then a result is determined. If a flag is set by any operation and the corresponding mask bit is 0, then an exception is generated and no value is written to the destination for any operation. Otherwise, the result that has been determined for each operation is written.

These three phases and the pre- and post-computation SSE exceptions are discussed in detail in the following sections.

12.3 Pre-computation Exceptions

The first step in the execution of any SSE floating-point instruction, before any exception checking is performed, is to examine the DAZ bit of MXCSR. If this bit is set, then any denormal operand is replaced by a zero of the same sign.

The conditions that may cause an exception flag to be set, or the operation to be terminated with a QNaN value, or both, prior to an SSE computation are as follows:

- SNaN operand: IE is set and the operation is terminated. If the first NaN operand is a QNaN, then the value is that operand; if the first NaN operand is an SNaN, then the value is that operand converted to a QNaN. For this purpose, in the case of an FMA $a \cdot b + c$, the operands are ordered as a, b, c.
- QNaN operand and no SNaN operand: No flag is set, but the operation is terminated. The value is the first NaN operand.
- Undefined Operation: IE is set, the operation is terminated, and the value is the real indefinite QNaN (Definition 5.23). The operands for which this condition holds depends on the operation:
  - Addition: Two infinities with opposite signs;
  - Subtraction: Two infinities with the same sign;
  - Multiplication: Any infinity and any zero;
  - Division: Any two infinities or any two zeroes;
  - Square root extraction: Any operand with negative sign, excluding negative zero;
  - Multiply-accumulate: A product of an infinity and a zero (with no restriction on the other operand), or a product of an infinity and any non-NaN added to an infinity with sign opposite to that of the product.

Exception or termination condition	Flag set	QNaN result (masked case)
SNaN operand	IE	QNaNized operand
QNaN operand	None	Operand
Undefined operation	IE	Indefinite QNaN
Zero exception	ZE	None
Denormal operand	DE	None

Table 12.2 SSE pre-computation exceptions

- A division operation with any zero as divisor and any finite numerical dividend: ZE is set, but the operation proceeds (resulting in an infinity) unless an unmasked exception occurs.
- Any denormal operand (with DAZ = 0) and none of the above conditions: DE is set, but the numerical computation proceeds unless an unmasked exception occurs.

Note that these conditions are prioritized in the order listed: if any condition holds for a given operation, then any other of lower priority is ignored for that operation. For a packed instruction, all operands are examined in parallel for pre-computation exception conditions. Consequently, it is possible for different flags to be set for different operations.

If any exception flag is set during this process and the corresponding mask bit is clear, then all operations are terminated before any computation is performed, no result is written to the destination, and an exception is generated.

If an operation of a packed instruction is terminated with a default value, the value is not written to the destination until execution of the instruction is completed, since no value is written in the event of an unmasked post-computation exception.

The setting of status flags and the default values are summarized in Table 12.2.

12.4 Computation

Unless terminated in response to a pre-computation exceptional condition, each operation of an SSE arithmetic instruction computes an unrounded value, which is then processed according to the contents of the MCXSR and the floating-point format of the instruction as described below. In the case of a packed instruction, all operations for which a default QNaN value has not been determined in the pre-computation stage are similarly processed in parallel.

For each of these operations, a value is computed. If this value is infinite or 0, then no flags are set and the sign of the result is determined by the signs of the operands and the rounding mode $\mathcal{R} = \text{MXCSR}[14:13]$:

- Infinity: The result is an infinity with sign determined according to the operation:
  - Addition: The sign of the infinite operand or operands.
  - Subtraction: The sign of the minuend if it is infinite, and otherwise the inverse of the sign of the subtrahend.
  - Multiplication or division: The product (xor) of the signs of the operands.
  - Square root: The sign of the operand, which must be positive.
  - Multiply-accumulate: The sign of the addend if it is infinite, and otherwise the product (xor) of the signs of the factors.
- Zero: The result is a zero with sign determined according to the operation:
  - Addition: The sign of operands if they agree; if not, then negative if $\mathcal{R} = RDN$, and otherwise positive.
  - Subtraction: The sign of the minuend if it is the inverse of that of the subtrahend; if not, then negative if $\mathcal{R} = RDN$, and otherwise positive.
  - Multiplication or division: The product (xor) of the signs of the operands.
  - Square root: The sign of the operand.
  - Multiply-accumulate: The product of the signs of the factors if it agrees with that of the addend; if not, then negative if $\mathcal{R} = RDN$, and otherwise positive.

Otherwise, execution proceeds to the next phase with the unrounded computed value, which is finite and nonzero.

12.5 Post-Computation Exceptions

The procedure described in this section is applied in the same way to all operations under consideration. For each operation that reaches this phase, the precise mathematical result (which, of course, need not be computed explicitly by an implementation) is a finite nonzero value u, which is rounded according to the rounding mode $\mathcal{R}$ and the precision p of the data format F, producing a value $r = rnd(u, \mathcal{R}, p)$. This value is subjected to the following case analysis, which may result in the setting of one or more exception flags. If any operation produces an unmasked exception, no result is written for any operation. Otherwise, a final result is written for each operation.

- Overflow (r is above the normal range of the target format, i.e., $|r| > lpn(F)$): In all cases, OE is set.
  - Masked Overflow (OM = 1):

    PE is set. The final result, which is valid only if PM = 1, depends on $\mathcal{R}$ and the sign of r.

    If (a) $\mathcal{R} = RNE$, (b) $\mathcal{R} = RUP$ and $r > 0$, or (c) $\mathcal{R} = RDN$ and $r < 0$, then the final result is an infinity with the sign of r.

Otherwise, the result is the encoding of the maximum normal value for the target format, $\pm lpn(F)$, with the sign of r.

– Unmasked Overflow (OM = 0):
No final result is returned. If $r \neq u$, then PE is set.

- Underflow (r is below the normal range, i.e., $0 < |r| < spn(F)$):
  – Masked Underflow (UM = 1):
  If FTZ = 1, then UE and PE are set. The final result, which is valid only if PM = 1, is a zero with the sign of r.
  If FTZ = 0, then u is rounded again to produce $d = drnd(u, \mathcal{R}, F)$, which may be a denormal value, 0, or the smallest normal, $\pm spn(F)$. If $d \neq u$, then both UE and PE are set; otherwise, neither flag is modified. The final result, which is valid unless PE is set and PM = 1, is the encoding of d, with the sign of u if $d = 0$.
  – Unmasked Underflow (UM = 0):
  UE is set. No final result is returned. If the $r \neq u$, then PE is set.
- Normal Case (r is within the normal range, i.e., $spn(F) \leq |r| \leq lpn(F)$):
  If $r \neq u$, then PE is set. The final result, which is valid unless PE is set and PM = 1, is the normal encoding of r.

Thus, PE indicates either that the rounded result is an inexact approximation of an intermediate result, or that some other value has been written to the destination. In the case of masked underflow, it may not be obvious that the behavior described above is consistent with the definition of the auxiliary ACL2 function *sse-round*, which sets PE if either $r \neq u$ or $d \neq u$. However, Lemma 6.118 guarantees that if the first inequality holds, then so does the second.

Also note there is one case in which underflow occurs and UE is not modified: UM = 1 and the unrounded value is returned as a denormal.