Chapter 14
Arm Floating-Point Instructions

The first Arm *Floating-Point Accelerator*, which appeared in 1993, resembled the x87 coprocessor in its use of an 80-bit EP register file. This was succeeded by the *Vector Floating-Point* (VFP) architecture, which included 64-bit registers implementing the single-, double-, and half-precision data formats. The *NEON Advanced SIMD* extension later added 128-bit instructions for media and signal processing applications.

The elementary arithmetic floating-point instructions of the Arm architecture are similar to the corresponding SSE instructions, including both scalar and vector (SIMD) versions. The behavior described in this chapter is common to the VFP and NEON elementary arithmetic instructions. Both architectures provide a floating-point status and control register, *FPSCR*, similar to the SSE MXCSR. A significant difference between SSE and Arm, however, is that the FPSCR does not include exception masks—all instructions are effectively masked. Consequently, an Arm instruction always executes to completion and returns a data result, and there is no interaction between the component operations of a SIMD instruction of the sort described in Sect. 12.2. However, the FPSCR does contain *trap enable* bits corresponding to the flags. If the trap enable bit is set when an exceptional condition occurs, then upon of completion of the instruction, hardware passes control to a trap handler instead of setting the flag.

Since the operations of a SIMD instruction behave independently, we shall confine our attention to the behavior of scalar instructions. The specifications of the instructions presented in this section are formalized by a set of three ACL2 functions, which belong to the library `books/rtl` in the ACL2 repository [13]:

- *arm-binary-spec*$(op, a, b, FPSCR, F)$
- *arm-sqrt-spec*$(a, FPSCR, F)$
- *arm-fma-spec*$(a, b, c, FPSCR, F)$

D. M. Russinoff, *Formal Verification of Floating-Point Hardware Design*,
https://doi.org/10.1007/978-3-319-95513-1_14

Each of these functions takes one or more operands (single-, double-, or half-precision floating-point encodings), the initial contents of the FPSCR register, and the data format F of the instruction, SP, DP, or HP. The function *arm-binary-spec*, which applies to the four binary arithmetic operations, takes an additional argument, *op*, representing the operation (*ADD*, *SUB*, *MUL*, or *DIV*). Each function returns two values: the data result and the final value of the FPSCR.

In Chaps. 16–19, we shall present a set of floating-point modules that have been implemented in an Arm processor. The above functions are the basis of the statements of correctness of these modules.

14.1 Floating-Point Status and Control Register

The FPSCR bits that are relevant to the instructions of interest are named as displayed in Fig. 14.1.

- Bits 4:0 and 7 are the *cumulative exception flags* for invalid operand (IOC), division by zero (DZC), overflow (OFC), underflow (UFC), inexact result (IXC), and denormal operand (IDE),
- Bits 12:8 and 15 are the *trap enables* corresponding to the flags, which determine whether, in the event of exceptional condition, the flag is set by hardware or control is passed to a trap handler. On an implementation that does not support exception trapping, these bits are held at 0.
- Bits 23:22 form the *rounding control* field (RC), which encodes a rounding mode as displayed in Table 14.1. Note the difference between this encoding and that of Table 12.1.
- Bit 24 is the *force-to-zero* bit (FZ), which, if set, coerces both denormal inputs and (except in the half-precision case—see Sect. 14.3) denormal results to ± 0. Thus, this bit plays the roles of both the DAZ and FTZ bits of the SSE MXCSR (Sect. 12.1).
- Bit 25 is the *default NaN* bit (DN). If asserted, any NaN result of an instruction is replaced by the real indefinite QNaN (Definition 5.23).

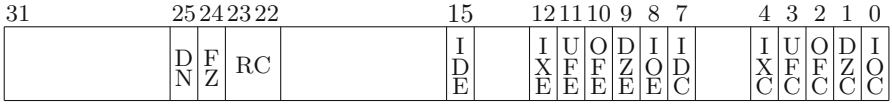

Fig. 14.1 FPSCR: Arm floating-point status and control register

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

Table 14.1 Arm rounding control

14.2 Pre-computation Exceptions

The floating-point pre-computation behavior of the Arm architecture is formalized by three functions:

- *arm-binary-pre-comp*$(op, a, b, FPSCR, F)$
- *arm-sqrt-pre-comp*$(a, FPSCR, F)$
- *arm-fma-pre-comp*$(a, b, c, FPSCR, F)$

Each of these returns an optional data value and an updated FPSCR. If a data value is returned, then execution is terminated; otherwise the computation proceeds.

We have noted that in the Arm architecture, a pre-computation exception never prevents the return of a value, and an exception flag is not set unless the corresponding trap enable is clear. The other departures from SSE pre-computation exception handling are in the setting of the denormal flag, the returned value in the case of a NaN operand, and the precedence of an undefined operation over a QNaN operand. Note, however, that the only case in which an undefined operation and a NaN operand can simultaneously occur is an FMA operation with a product of an infinity and a zero with a NaN addend.

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 Arm floating-point computation are as follows:

- Denormal operand: If FZ = 1, then the operand is forced to ± 0 and, unless the format is *HP*, IDC is asserted; otherwise, neither of these actions is taken. The setting of other flags is based on the result of this step. Note that a denormal exception is not suppressed by another exceptional condition; it is possible for two flags to be set.
- SNaN operand: IOC is set. If DN = 1, then the real indefinite QNaN is returned (Definition 5.23). Otherwise, the first SNaN operand is converted to a QNaN and returned. For this purpose, in the case of an FMA $a + b \cdot c$, the operands are ordered as a, b, c.
- Undefined Operation: The conditions are as specified in Sect. 12.3. IOC is set and the real indefinite QNaN is returned.
- QNaN operand and no SNaN operand or undefined operation: If DN = 1, then the real indefinite QNaN is returned. Otherwise, the first NaN operand is returned (with FMA operands ordered as in the SNaN case).
- A division operation with any zero as divisor and any finite numerical dividend: IDZ is set, but the operation proceeds (resulting in an infinity).

14.3 Post-Computation Exceptions

If a final result is not produced during the pre-computation phase, then control is passed to one of the following, which computes an unrounded value:

- *arm-binary-comp*$(op, a, b, FPSCR, F)$
- *arm-sqrt-comp*$(a, FPSCR, F)$
- *arm-fma-comp*$(a, b, c, FPSCR, F)$

If the computed value is infinite or 0, then execution is terminated. No flags are set and the sign of the result is determined by the signs of the operands and the rounding mode $\mathcal{R} = \text{FPSCR}[23 : 22]$ as described in Sect. 12.4.

Otherwise, the precise mathematical result of the operation is a finite nonzero value u. This value is passed to the common function

$$arm\text{-}post\text{-}comp(u, FPSCR, F),$$

which performs the rounding and detects exceptions as described below. Note that in addition to the absence of exception masks, there are several departures from SSE behavior in the detection and handling of underflow.

Unless u is a denormal, it is rounded according to the rounding mode $\mathcal{R}$ and the precision p of the data format f, producing a value $r = rnd(u, \mathcal{R}, p)$. The returned value and the setting of exception flags are determined by the following case analysis. In all cases, the setting of a flag is understood to be contingent on the value of the corresponding trap enable bit, except in a certain case of underflow as noted below.

- Overflow (r is above the normal range of the target format, i.e., $|r| > lpn(F)$):
  In all cases, OFC and IXC are set. The result 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.
- Underflow, which is detected before rounding (u is below the normal range, i.e., $0 < |u| < spn(F)$):
  If FZ = 1 and F is *SP* or *DP*, then UFC is set. (IXC is not set.) UFE is ignored in this case, and FZ is ignored if F is *HP*. The final result is a zero with the sign of u.
  If FZ = 0 or F is *HP*, then u is rounded 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 UFC and IXC are set; otherwise, neither flag is modified. The final result is the encoding of d, with the sign of u if $d = 0$.
- Normal Case (u and r are both within the normal range):
  If $r \neq u$, then IXC is set. The final result is the normal encoding of r.