Floating-Point Units

Floating-point mathematical operations are required in a wide range of embedded applications, such as control systems and digital signal analysis. Where an application does require floating point, the performance targets often dictate the need for a hardware-based floating-point unit. Most embedded processor SOCs offer a version that provides hardware-based floating point. A key attribute of a floating point acceleration function associated with the processor is whether the floating-point unit is compliant with the IEEE Standard 754 for Binary Floating-Point Arithmetic. The precision of the floating-point unit (single/double) is also an important attribute in developing the floating-point algorithms.

[Intel processors](https://www.sciencedirect.com/topics/computer-science/intel-processor) have two floating-point units. The first and probably best known is the x87 Floating-Point Unit (FPU). The x87 FPU instructions operate on floating-point, integer, and binary-coded decimal (BCD) operands. It supports 80-bit precision, double extended floating-point. The FPU operates as a [coprocessor](https://www.sciencedirect.com/topics/computer-science/coprocessor) to the instruction unit. The FPU instructions are submitted to the FPU and the scalar (main processor) pipeline continues to run in parallel. To maximize overall application performance it is important to ensure that the processor’s main execution flow can perform useful work while the FPU performs the floating-point operations. To that end it is usually best to use a compiler to generate the target code to ensure efficient instruction scheduling.

Not all floating-point operations can be completed; for example, dividing a number by zero results in a floating-point fault. The following floating-point operations result in faults: all invalid operations, for example, square root of a [negative number](https://www.sciencedirect.com/topics/computer-science/negative-number), overflow, underflow, and inexact result. The operating system provides an [exception handler](https://www.sciencedirect.com/topics/computer-science/exception-handler) to handle the floating-point fault exceptions. In the case of Linux, the kernel catches the fault and sends a user space signal (SIGFPE, signal floating-point exception). The application will be terminated unless the application has chosen to handle the exceptions. The C language specification as defined by the ISO C99 (ISO/IEC 9899:1999) standardizes a number of functions to control the behavior of floating-point rounding and exception handling. One such function is fegetexceptflag().

Intel processors also have a [Single Instruction Multiple Data](https://www.sciencedirect.com/topics/computer-science/single-instruction-multiple-data) (SIMD) execution engine. The [Intel Atom processor](https://www.sciencedirect.com/topics/computer-science/intel-atom-processor) supports the Supplemental Streaming [SIMD](https://www.sciencedirect.com/topics/computer-science/single-instruction-multiple-data) Extensions 3 (SSSE3) version of the SIMD instructions, which support integer, single, and double precession floating-point units.

The Intel Atom processor has a rich set of floating-point coprocessor capabilities. As a result, a particular algorithm could be implemented in a number of ways. The trade-offs for the use of each floating-point unit for a particular operation are described in Chapter 11, “Digital Signal Processing.”

The floating-point units are resources that contain a number of registers and some current [state information](https://www.sciencedirect.com/topics/computer-science/state-information). When different software threads wish to use the resources, the kernel software may have to save and restore the registers and all state information during operating system context switches. The Intel processor provides FXSAVE and FXRSTOR to save and restore the required state and register information of FP and SSE units. These operations can be costly if performed on every task transition, so Intel processors provide a mechanism to help the kernel identify whether an FPU was actually used by a particular process. The TS flag in the control register zero (CS0.TS) provides an indication that the floating point unit has been used. The kernel can clear the value when it performs a context switch, and check if the bit has been set during the execution of the process (indicating the process used the FP unit). The operating system can be configured to save the registers and state on transition from a thread that used the resource or alternatively raise an exception when a new thread attempts to use the resource after a previous thread has used it. If the real-time behavior of your FP/SSE code is important, you should look into the detailed operating [system behavior](https://www.sciencedirect.com/topics/computer-science/system-behavior). You may have to take special steps if you want to use floating-point or SSE units from within the kernel. For example, in Linux you have to call kernel\_fpu\_begin() and kernel\_fpu\_end() around the code that uses the FP/SSE units. This will save and restore the required state.