Section 2 / Floating Point

This chapter deals exclusively with the handling of floating point operations on the AARCH64 platform.

What are Floating Point Numbers?

Let's first begin with an understanding of what floating point numbers are. That can be found here.

The TL;DR is that floating point numbers are approximations with double precision being better approximations that single precision.

Floating point numbers have a sign bit (for signed floating points), an exponent which controls the range of representable numbers, and a mantissa that controls precision.

Floating Point Registers

There are 31 general registers (the X and W registers). Similarly there are 31 floating point registers which are reused for single, double and vector (SIMD - Single Instruction Multiple Data) instructions.

A bit more detail is provided here.

Rounding and Truncation

Truncation is part of casting `float` and `double` to `int` and `long`.

Rounding is important too.

Coverage on rounding and truncation is found here.

Loading Floating Point Numbers into Registers

This is a little confusing because some values can be loaded from arguments in the `fmov` instruction. For example, 1.0 can be `fmov`ed. Trying to do the same for 1.1 will fail. Remember that AARCH64 instructions are always 32 bits wide and that floating point numbers are at least that size.

This chapter covers the loading of floating numbers into registers. A sample program is linked below.

Nuances of `fmov`

As indicated above, you can `fmov` a floating point literal into a register. Except when you can't. Well, mostly you can't.

Additionally, there are some rules about using `fmov` between registers.

This chapter covers the nuances of using `fmov`.

Half Precision Floating Point Numbers

Often used in Computer Graphics, half precision floats fit within 16 bits, the size of a `short`.

The TL;DR here is: Avoid them.

This chapter explains why.

SIMD

There are two types of SIMD instruction sets available in the AARCH64 ISA but the makers of processors are not obligated to implement them on any particular chip.

The first kind is NEON. This is described here.

The second kind of Scalable Vector Extension (SVE) for which we do not have near-term plans to cover. SVE is not implemented on any generally available ARM processor including Apple Silicon.

Demo Programs in this Chapter

In case you want to get right to the code, here are the demos presented in this chapter.

If you receive the assembly language files with a lower case extension, you will need to make the `.s` extension into `.S`.

Link	Contents	Converged
Link	Deconstructs floating point values	NA
Link	Demonstrates some rounding in asm	Yes
Link	Demonstrates some rounding in C++	NA
Link	Demonstrates dealing with floating point literals	Yes