fmov

The `fmov` instruction is used to move floating point values in and out of floating point registers and to some degree, moving data between integer and floating point registers.

Loading Floating Point Numbers as Immediate Values

Just as we saw with integer registers, some values can be used as immediate values and some cannot.

For example, this works:

```
mov    x0, 65536
```

but this does not:

```
mov    x0, 65537
```

The reason is that all AARCH64 instructions must fit within a 32 bit instruction that must hold the instruction's op code, its flags and other bits and bobs plus any immediate value. In the above example we can see that the `mov` instruction provides up to 16 bits for an immediate value.

The constraints placed on immediate values for `fmov` are much tighter because floating point numbers are far more complex than integers.

Make sure you have read and understand this chapter before proceeding.

Let's take a look at some code:

```
fmov        d0, 1.0         // works
fmov        d0, 1.5         // works    2**-1
fmov        d0, 1.75        // works    2**-1 + 2**-2
fmov        d0, 1.875       // works    2**-1 + 2**-2 + 2**-3
fmov        d0, 1.9375      // works    the preceding + 2**-4
fmov        d0, 1.96875     // Zoinks!
```

From this we can see that an immediate value for an `fmov` seems to have 4 bits available for the mantissa. In fact, the only values that work as immediate values will be those floating point values whose fractional values are combinations of:

- 1/2
- 1/4
- 1/8 and
- 1/16

As far as exponents go, `fmov` can accommodate 3 bits. So, exponents of plus or minus 2**7 can be used.

Loading / Storing Floating Point Numbers in General

When in doubt, load fixed floating point numbers from memory. This is covered in this chapter.

SIMD

`fmov` can also deal with the more complicated special cases induced by SIMD instructions.

Movement To / From Integer Registers

`fmov` can *bits* between the integer and floating point registers. We emphasize the *bits*. No conversions are done using `fmov`. There exist other instructions for that. See this chapter for more information.