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. It comes down to how many bits are necessary to encode the value. Too many bits... not enough room to fit in a 4 byte instruction plus the opcode.

For example, this works:

```
mov    x0, 65535
```

but this does not:

```
mov    x0, 65537
```

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` has 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.

A sign bit makes the total number of bits available for immediate moves to be 8.

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. `fmov` is able to move values between the various register widths such as single precision to double precision. **However, no conversion of value is performed - `fmov` just copies bits.**

If you need to change the precision of a floating point value, the `fcvt` family of instructions must be used instead.

Movement To / From Integer Registers

`fmov` can copy *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.