2.10 Floating-point representation

You have probably already have met standard scientific notation; e.g.

$$6.023 \times 10^{23}$$

- Here a number is represented as an integer place followed by a number of significant digits.
- We can do this in binary by simply moving the binary point. (commonly referred to as the decimal point) to a position to maximise the number of significant (digits) bits. Then the position of the binary point is recorded in an exponent representation
- Hence the name *floating point*; Here one plays off *efficiency* with *range* and *accuracy*.
- Since the Real numbers are continuous, we can only approximate these in a computer. Floating point maximises the resolution within a given set of space constraints.
- *Floating point* is in fact a form of sign magnitude. Actually *sign exponent magnitude*.

As a general rule for binary

$$\pm 0.f \times 2^{\pm e}$$

or more formally write

$$X = (-1)^S \times fraction \times 2^{\{exponent-K\}}$$

2.10.1 Sign S

0 for a positive number, 1 for a negative number.

2.10.2 Fraction

Just like you are used to, e.g.,

.0100011010

- The accuracy depends on number of bits. The idea is to maximise the number of significant bits, using the exponent to record the position of the point.
- We can always shift (except in the case of zero) till the MSB is a 1. Thus we can assume MSB to be a 1 (or 0) and then save space by not showing it.

- This gives rise to the *normalised fraction* $0.5 \leq f \leq 1.0$

  e.g.

  $.10000000_2$ 1/512 absolute error (.0019) : 0.4%
  $.11111111_2$ 1/512 absolute error: 0.2%

2.10.3 Exponent

The convention is to use the Excess-K notation.

- To explain this further it is appropriate to use one of the VAX formats.

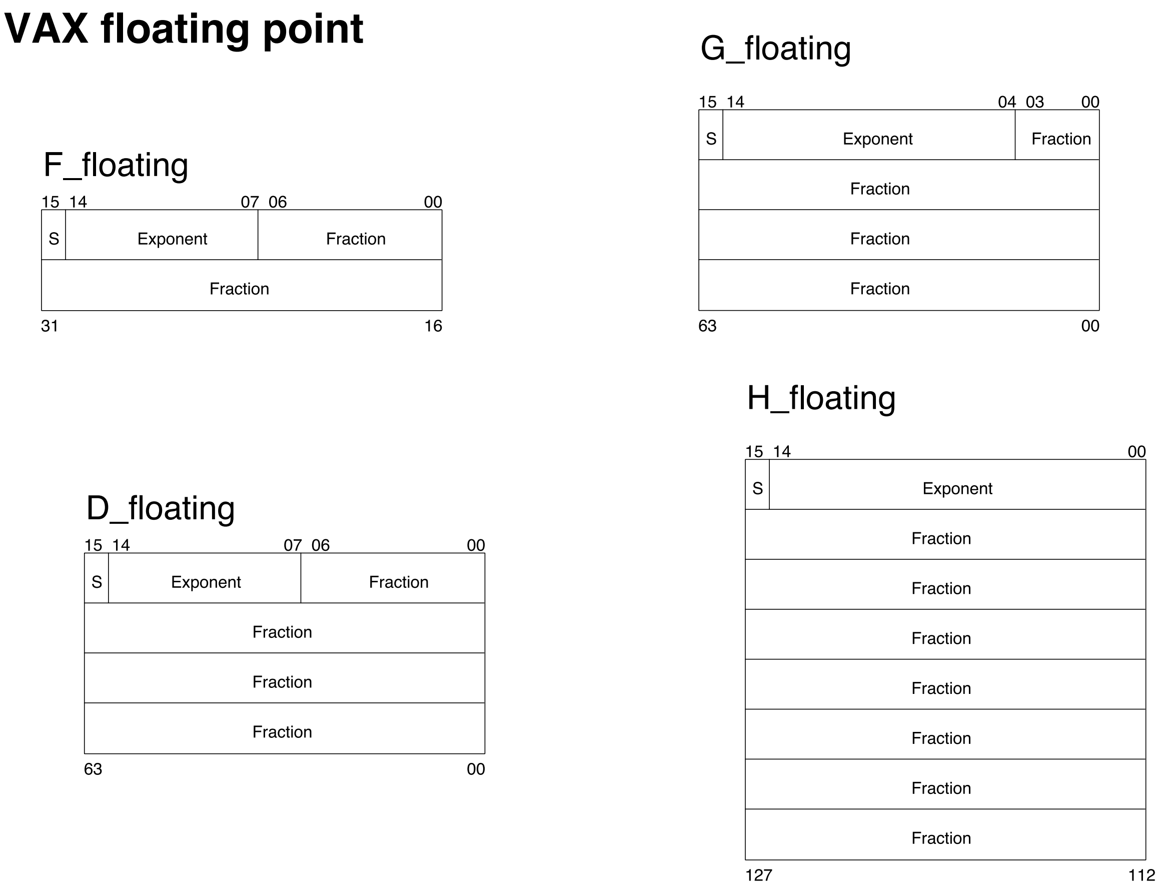
2.11 The VAX formats

The Alpha supports amongst its various data representations, the vax floating point formats. These are designated;

- F (Floating, Single precision, 32 bits, 2 words, 4 bytes)
- D (Double precision, 64 bits, 4 words)
- G (Grand 64 bits)
- H (Huge 128 bits 8 words, quadruple precision)

(The sign appears on the 16-bit word boundary which is an historic artifact reflecting the days of the PDP-11).

VAX data types supported on the ALPHA.



Take the G_floating format for example:

Sign (0/1 S)
Note the position of the sign on the 16th bit boundary. This is a hang over from the days of the PDP-11.

Fraction/mantissa (0.1M)
bits $0-3 = 4$ bits $+3 \times 16$ bits $= 52$ bits plus the implied bit $(.1) = 53$ bits covering $0.5 \leq fraction < 1.0$ normalised.

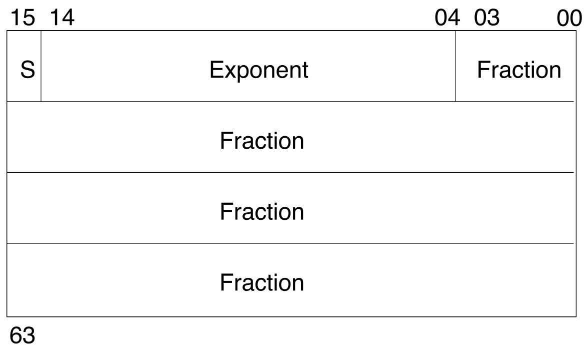
Exponent (XS-1024)
In the 11-bit notation, excess 1024 yields the range of values $1-2047$ i.e. of -1023 to $+1023$.

A 0 exponent together with a 0 sign indicates a value 0 irrespective of the value of mantissa/fraction.

Example 2.11.1 *Suppose we have a G_Floating number*

$$0000000018004080_{16}$$

and we want to convert this to decimal;



Sign *is 0 so we know the number is a positive number.*

Exponent $408_{16} = 1032_{10}$
Offset by 1024*, so exponent is* $1032_{10} - 1024_{10} = 8$.

Fraction
Re-constitute the fraction 0180000000000_{16}

in binary $000000011000\ldots\ldots0000_2$

normalised form;

so add a 1 *on the left with the binary point*

$\underline{.1}000000011000\ldots0000$

Final conversion: *multiply by the exponent* 2^8

$$10000000.11$$

$$128 + .5 + .25 = 128.75_{10}$$