The Character of Binary8 Floating-Point Formats

Jeffrey Sarnoff

2023-Oct-24 (v0.6.7)

Early pre-release for review and corrections. This version may contain errors or inaccuracies.

Bits Spanned

k is the number of bits required of the binary interchange format (always 8 for binary8s)

# an *octothorpe* '#' starts an inline comment

```
const k  = 8  # the bits required to encode binary8 values
const kₘ = 7  # the bits required to encode binary8 magnitudes[1] (k − 1)
```

All binary8 formats have 256 == 2^8 == 2^k unique encodings. Binary8 formats are given uniquely by their precision. Other binary floating-point interchange formats may be developed using the parametric expressions provided by changing the value of k to any other value in [3, 15].

Each binary8 format maps the same 256 encodings (0x00..0xff) onto its own set of 256 values. Each of these values is unique within its value set. Distinct binary8 formats may share some of their values. Where subnormal or normal values are shared by formats, they are mapped from distinct encodings.

Precision

p is the precision of the format in bits (p includes the implicit bit of the significand). The number of bits explicitly stored for the significand is precision – 1. This has to be >= 0; so, the smallest precision is 1. The largest precision for binary8 formats is 8, which is used with a 7-bit wide explicit significand. There is always a sign bit, so 7 bits is the most available for other purposes.

The name binary8p{ **p** } selects an 8-bit binary floating-point interchange format (binary8 format) by the precision **p** of the format. Valid formats are binary8p1, binary8p2, binary8p3, binary8p4, binary8p5, binary8p6, binary8p7, binary8p8. Binary8p8 provides integer-valued floats. Binary8p1 supports a wide range of values (0.0, 2e-19 .. 0.5, 1.0, .. 9e+18, +Inf).

[1] k_m is not a formal parameter from 754. It may be used to simplify other parametric expressions.

Kinds of Value

Values are either Normal, Subnormal or Special. These are the 4 special values {0, +Inf,-Inf, NaN}. Special values are neither normal nor subnormal as 754 defines those terms. There are 252 (256-4) ordinary values, each ordinary value is either a normal or a subnormal value.

```
const n_values        = 2^k                          # 256, encoding count
const n_specials      = 4                            # special value count
const n_ordinaries    = n_values - n_specials        # 252, ordinary count
```

signed and unsigned values

Binary8 formats have n_values. Half of them are encoded with the msb[2] clear (nonnegative values), and half are encoded with the msb set (negative values). Note that half of our four special values are nonnegative, and half are negative. Half of the ordinary values are positive, half negative.

With binary8 formats, zero is neither negative nor positive; consider zero unsigned until there is a compelling reason to ascribe some oriented sense, then use the positive sense. For example, evaluating divide(0x01, 0x00) is +Inf rather than-Inf or NaN.

```
const n_pos_ordinaries = n_ordinaries >> 1           # 126
const n_pos_numbers    = n_pos_ordinaries + 1        # 127, includes +Inf
const n_nonneg_numbers = n_pos_numbers + 1           # 128, includes Zero
```

magnitudes

Ordinary values are signed values. They are positive (the msb is 0b0) or negative (the msb is 0b1)

There are (n_ordinaries >> 1) positive values and (n_ordinaries >> 1) negative ordinary values.[3]

Where convenient, we let strictly positive numbers serve as nonzero magnitudes. Note that Zero and +Inf,-Inf are not magnitudes. Each is neither a normal nor a subnormal value (*matching their status in 754*). This approach works by relying on the msb of a positive value to be 0b0. That is true everywhere, throughout the specification of binary8 formats.

```
const n_ordinary_mags[4] = n_ordinaries >> 1# 126, subs+normals / 2
```

---

[2] *msb* is an acronym for *most significant bit*

[3] With positive evens, div by 2 is (pos_even >> 1). This form abstracts away language specifics in (pos_even / 2).

[4] “mags” abbreviates “magnitudes”

Parameters

IEEE Std 754-2019

w	bitwidth of exponent field	derived from k, p	8 - p
t	explicit significand bits	derived from p	p - 1
emax	max unbiased exponent	defined from k_m, p	2^(7 - p) - 1
bias	exponent bias, emax(p) + 1	defined from k_m, p	2^(7 - p)

mnemonics

```
sig_bits(p) = p                            # the precision
frc_bits(p) = p - 1                        # the fractional part

exp_bits(p) = 8 - p                        # k - p, field bits
exp_bias(p) = p<8 ? 2^(7 - p) : 0          # 2^(exp_bits(p) - 1)
exp_max(p)  = p<8 ? 2^(7 - p) − 1 : 0      #  exp_bias(p) - 1
```

Counts

The binary8p1 format has normal values only. The binary8p8 format has subnormal values only. All other binary8 formats have subnormal values and normal values.

Counting Significands

Significands do not incorporate the sign bit. The number of unique significands for normal values of a binary8 format is 2^(precision − 1). Subnormal values do not include the zero-valued significand (that is used to encode Zero and NaN). Binary8p8 has no normal values, and so has no normal significands.

```
n_normal_significands(p)    = 2^(p - 1) % 2^7
n_subnormal_significands(p) = 2^(p - 1) - 1 - (p >> 3) # min(126, 2^(p-1) - 1)
```

Counting Exponents

The number of unique exponents available in a binary8 format is 2^exp_bits(p). Subnormal values use one exponent, the exponent corresponding to a zeroed exponent bitfield. Normal values use all available exponents, including the exponent used by subnormal values. Binary8p8 has no normal exponents.

```
n_normal_exponents(p)    = 2^(8 - p) - 1
n_subnormal_exponents(p) = (p > 1) ? 1 : 0             # 1 - (p == 1)
```

Counting Values

There is one subnormal value for each subnormal significand. There is exactly one subnormal exponent; subnormal significands do not cycle. “mags” abbreviates “magnitudes”.

```
n_subnormals(p)     = p>1 ? 2^(p)-2–2*(p>>3) : 0    # n_normals(p)-2
n_subnormal_mags(p) = n_subnormals(p)>>1
```

The number of normal values is given from n_ordinaries and the number of subnormal values. There are half as many normal magnitudes as there are normal values.

```
n_normals(p)     = p<8 ? n_ordinaries-n_subnormals(p) : 0
n_normal_mags(p) = n_normals(p)>>1
```

Extremal Magnitudes[5]

exponents

```
min_exponent(p)      = p<8 ? 1 - 2^(7-p) : 1         # 1-exp_bias(p)
min_exponent_val(p)  = 2.0^(min_exponent(p))

max_exponent(p)      = p<8 ? 2^(7-p) - 1 : 0
max_exponent_val(p)  = p<8 ? 2.0^max_exponent(p) : 1.0
```

significands

```
min_subnormal_sig(p) = recip(2^(p-1))

max_normal_sig(p)        = 1 + (2^(p)-1) // (2^p)
penult_max_normal_sig(p) = 1 + (2^(p)-2) // (2^p)    # sig before max
```

values

```
min_normal(p)        = (p<8) ? min_exponent_val(p) : 2.0
max_normal(p)        = penult_max_normal_sig(p) * max_exponent_val(p)

min_subnormal(p)     = min_subnormal_sig(p) * min_exponent_val(p)
max_subnormal(p)     = min_normal(p) - min_subnormal(p)
```

[5] recip(x) is 1/x. This indirection is used so rational values can be obtained independent of the way ‘/’ evaluates.

End Page