.۱

کوچک‌ترین عدد مثبت هنگامی است که قسمت اعشار (F_{min}) کمترین مقدار را داشته باشد و مقدار نما کوچکترین مقدار را داسته باشد (E_{min}) و بیت علامت صفر باشد.

در فرمولی که برای نشان دادن مقدار عدد استفاده میشود، عبارت $(2 \times b_{31} - 1)$ برای بیت علامت است (اگر b_{31} صفر باشد، عدد منفی و در غیراینصورت عدد مثبت میشود) و عبارت (2^{E-64}) برای قسمت نما است، برای بدست آوردن کوچکترین نما:

$$E_{min} = \Sigma_{i=24}^{30}\left(2^{i-24} \times b_i\right) = \Sigma_{i=24}^{30}\left(2^{i-24} \times 0\right) = 0$$

عبارت $\left(\Sigma_{i=0}^{23}\left(\overline{b_i} \times 2^{i-12}\right)\right)$ برای مقدار اعشار است، که کوچکترین مقدارش برابر است با:

$$F_{\min} = 1 \cdots 10_{binary} = 2^{-12}$$

پس کوچکترین عدد مثبت برابر است با:

$$2^{0-64}(2 \times 1 - 1)(2^{-12}) = 2^{-76} \xrightarrow{\text{in binary}} 1\ 000\ 0000\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110$$

کوچکترین عدد منفی دارای بزرگترین نما و بزرگترین اعشار و علامت کلی منفی است:

$$E_{\max} = \Sigma_{i=24}^{30}\left(2^{i-24} \times 1\right) = 2^6 + 2^5 + \cdots + 2^0 = \frac{(1)(1-2^7)}{1-2} = 127$$

$$F_{\max} = \Sigma_{i=0}^{23}\left(1 \times 2^{i-12}\right) = 2^{11} + 2^{10} + \cdots + 2^{-11} + 2^{-12} = \frac{(2^{-12})(1-2^{24})}{1-2} = 2^{12} - 2^{-12}$$
$$= 4095.99975586$$

پس کوچکترین عدد منفی برابر است با:

$$2^{127-64}(2 \times 0 - 1)(4095.99975586) = 2^{51} - 2^{75}$$
$$= -4095.99975586 \times 2^{63} \xrightarrow{\text{in binary}} 0\ 1111\ 1111\ 0000 \ldots 0000$$

۲.

$$2AE44200 = 00111010111001001000010000000$$
$$= +(2^{-1} + 2^{-2} + 2^{-5} + 2^{-9} + 2^{14}) \times (2^{-42})$$

$$121.0121 = 01000010111100100000011000110010$$

۳.

الگوریتم جمع/تفریق اعداد اعشاری

۱. چک کردن صفر

۲. مقایسه نماها (جهت پیدا کردن عدد با نمای بزرگتر)

۳. عدد با نمای کوچکتر را به اندازه نماها به سمت راست شیفت می‌دهیم. } ردیف کردن نما

۴. دو عدد را با هم جمع/تفریق می‌کنیم (جمع/تفریق کننده اندازه علامت)

۵. در صورتی که حاصل هنجار نباشد؛ آن را هنجار می‌کنیم.

$$0.75 = 1.1 \times 2^{-1}, (-0.4375)_{10} = (-1.11)_2 \times 2^{-2}$$

$$-0.111 \times 2^{-1} + 1.1 \times 2^{-1} = 0.101 \times 2^{-1} = 1.01 \times 2^{-2}$$

الگوریتم ضرب اعداد اعشاری

۱. چک کردن صفر

۲. جمع نماها با هم $E_R = E_A + E_B - b$

۳. تعیین علامت حاصل $S_R = S_A \oplus S_B$

۴. انجام عمل ضرب $X = 1.F_A \times 1.F_B$

۵. هنجار کردن نتیجه در صورت نهنجار شدن (overflow)

.۴

سرریز[۱] پس از انجام عملیات عدد حاصل شده، قبل از ممیز عددی بزرگتر از یک داشته باشد، مثل:

$$1.0011 \times 2^{10} + 1.0010 \times 2^{10} = 10.0001 \times 2^{10}$$

زیرریز[۲] قبل از ممیز فقط صفر وجود داشته باشد، در ضرب نداریم. مثل:

$$-0.4375_{10} + 0.5_{10} = (-1.110)_2 \times 2^{-2} + (1.000)_2 \times 2^{-1}$$
$$= (-0.111)_2 \times 2^{-1} + (1.000)_2 \times 2^{-1} = (0.001)_2 \times 2^{-1}$$

بی‌اعتبار[۳] اگر عملیاتی که نوشتیم برای داده تعریف نشده باشد، این پرچم روشن می‌شود. مثل: $\sqrt{-2}$

یا مقایسه با *Nan*:

روشن شدن پرچم بی اعتباری در مقایسه با *nan*	مقایسه ها
No	*== or !=*
Yes	*< or <= or > or >=*

نادرست[۴] عملیات نتیجه ای میدهد که نمیتوان با دقت بینهایت نشان داد. مثلک

$$\frac{2.0}{3.0} \text{ or } \log(1.1)$$

تقسیم‌برصفر[۵] مقدار بینهایت تولید میشود و هنگام تقسیم عدد غیرصفری بر صفر ایجاد میشود. مثل:

$$\frac{9.0}{0.0}$$

۵.

Type	Sign	AE	BE	EF	FF	V
Zero	0	-126	0	0000 0000	000 0000 0000 0000 0000 0000	0.0
Negative Zero	1	-126	0	0000 0000	000 0000 0000 0000 0000 0000	-0.0
One	0	0	127	0111 1111	000 0000 0000 0000 0000 0000	1.0
Minus One	1	0	127	0111 1111	000 0000 0000 0000 0000 0000	-1.0
Smallest denormalized number	*	-126	0	0000 0000	100 0000 0000 0000 0000 0000	2^{-23} $\times 2^{-126}$
0Largest denormalized number	*	-126	0	0000 0000	111 1111 1111 1111 1111 1111	$(1-2^{-23})$ $\times 2^{-126}$
Smallest normalized number	*	-126	1	0000 0001	000 0000 0000 0000 0000 0000	(2^{-126})
Largest normalized number	*	127	254	1111 1110	111 1111 1111 1111 1111 1111	$(2-2^{-23})$
Positive infinity	0	128	255	1111 1111	000 0000 0000 0000 0000 0000	inf