4. Binary Floating-Point Numbers （浮動小数点数）

- 浮動小数点数表現
- 浮動小数点数の四則演算
- 浮動小数点数表現の選択
- IEEE 浮動小数点数の準拠
- 丸め手法

1.1 準備

浮動小数点数の表現は、次の 2 つで構成されている。仮数 M とべき指数 E である。組 (M, E) をもつ浮動点数 F は次の値を持つ。

$$F = M \cdot \beta^E$$

ここで β は、べき指数の基数である。この基数は与えられた系におけるすべての浮動小数点数に共通である。それゆえ、浮動小数点数の表現に含まれていないが、暗黙に使われている。

n ビット数表現の特徴

- 仮数 M と指数 E からなる。
- 表現できる範囲はより大きい。
- 異なる値の総数が 2^n なので、正確さは小さくなる。
- 連続した 2 つの数値の距離は大きい。

数表現の方式 :

- 仮数 M と指数 E は、両方とも符号付き。
- 2 つの仮数部の数表現は使用されている小数と範囲 $[1, 2)(for\ \beta = 2)$. における数。
- 1980 年まで、浮動小数点のための標準はなかった、
- 異なる機器の間のデータ伝送は難しかった。

仮数が符号ー絶対値小数である場合、浮動点形式は、符号ビット S、 e ビットの指数 E、 m ビットの符号無しの小数部 M から構成される。ここで $m + e + 1 = n$、次に示されるようになる。

S	指数 E	符号無し仮数 M

$$F = (-1)^S \cdot M \cdot \beta^E \qquad (4.1)$$

$(-1)^0 = 1$ と $(-1)^1 = -1$ から、小数の仮数部の最大値 M_{max} は、 $M_{max} = 1 - ulp$ に等しくなる。通常、 $ulp = 2^{-m}$。
基数 β は基数 $r = 2$ で整数のべき乗に制限する。すなわち、 $\beta = 2^k$ で $k = 1, 2, \cdots$。この理由は、同時に仮数を減らしたり、べき指数を増やしたりすることにより、浮動点数の値が変化しないからである。

算術演算は結果として $M_{max} = 1 - ulp$ の許される最大値より大きい仮数になれば、許される範囲内になるように仮数を減らさなければならない。同時に、浮動点数の値が同じままであるように、べき指数を増やさなければならない。いかなる基数 β の値において、次の単純な関係を持っている。

$$M \cdot \beta^E = (M/\beta) \cdot \beta^{E+1}$$

M/β における除算演算は、もし β が基数の整数べき乗であるとき、単純な算術右シフト演算になる。もし $\beta = r = 2$ ならば、その時は仮数を右に 1 シフトするのは、べき指数に 1 を加えることによって補正される。

例 4.1

算術演算の結果、M_{max} より大きい仮数で $01.10100 \cdot 2^{100}$ が得られた。これから、右に 1 つシフトすることによって仮数を減らすべきであり、また、$0.11010 \cdot 2^{101}$ にするために 1 だけ指数を増やすべきである。もし、$\beta = 2^k$ ならば、1 だけ指数を変えることが k 桁における仮数をシフトするのに相当する。すなわち、

$\beta = 4 = 2^2, \quad 01.10100 \cdot 4^{010} = 0.01101 \cdot 4^{011}.$

一般に、浮動点形式の数表現は複数ある。例えば、$0.11010 \cdot 2^{101} = 0.01101 \cdot 2^{110}$。111 のべき指数でも表現できるが、もし仮数部が 5 ビット長しかなかったら、仮数は 0.00110 になり、重要な桁はを失われてしまう。

仮数の最大数を保持するように正規化表現を使用する。正規化表現も浮動点数における比較を簡素化する。より大きなべき指数は、より大きな全体にわたる数を示している。そして、仮数は、等しい指数をもつもののみで比較されなければならない。

注意するのは、$\beta = 2^k$ の場合において、仮数は k 桁（または k の倍数）だけシフトすることができる。また、仮数はもし最初の k 桁に 0 でないビットがあれば正規化を考慮する。例えば、数 $0.00000110 \cdot 16^{101}$ の正規化表現は、$0.01100000 \cdot 100$ であり、1 つの先頭の 0 を消去できない。

もし分数が正規化されているならば、仮数の範囲は $[0, 1 - ulp]$ より小さくなる。許される最小値と最大値はその代わり次になる。

$$M_{min} = \frac{1}{\beta} \quad \text{and} \quad M_{max} = 1 - ulp$$

正規化された小数の範囲は、値 0 を含まないことに注意する。これゆえ、特別な表現が 0 のために必要になる。0 の可能な表現は $M = 0$ とべき指数 E で構成する。通常、浮動小数点における 0 の表現は固定点における表現と同じであり、0 命令の検査の実行を簡素化するためにも、$E = 0$ が選ばれる。

べき指数の表現方法は一般に偏倚 (偏り) べき指数を使用する。

$$E = E^{true} + bias$$

ここで、偏倚 (偏り) は常数で、E^{true} は 2 の補数で表現されたべき指数の真の値である。指数部の e ビットを使っている E^{true} の範囲は

$$-2^{e-1} \leq E^{true} \leq 2^{e-1} - 1$$

偏倚 (偏り) については通常、負のべき指数の最大絶対値として 2^{e-1} が選ばれる。

$$0 \leq E \leq 2^e - 1$$

この場合において、べき指数は 2^{e-1} 増し法で表現される。

この方法の利点は、2 つのべき指数の比較をするときに (加減算演算において必要とされるように)、符号無し数のように比較し、符号ビットを無視出来る。結果として、もし浮動点形式が S, E, M で構成されていれば、符号付き絶対値表現における 2 進整数のように浮動小数点数の比較ができる。もう一つの利点は、表現できる最小数が 0 と同じべき指数を持つと言うことである。

Example 4.2

$e = 7$ で 2 の補数表現におけるべき指数の範囲は、$-64 \leq E^{true} \leq 63$ となり、すなわち、$-64 = 1000000$ と $63 = 0111111$ となる。64 の偏倚 (偏り) を加える時に、-64 の真の値は 0000000 により表現される。また、63 の真の値は 1111111 により表現される。この表現は、64 増し法と呼ばれる。

2^{e-1} 増し表現は、2 の補数表現の符号ビットを単純に反転することにより得られる。符号ビットの値 0 と 1 が、それぞれ負数、正数を表している。正規化浮動点数の完全な範囲は、F^+ によって示される正の浮動点数、F^- によって示される負の浮動点数での同じ部分範囲を含んでいる。正の浮動点数の範囲は

$$M_{min} \cdot \beta^{E_{min}} \leq F^+ \leq M_{max} \cdot \beta^{E_{max}}$$

E_{min} が最小べき指数であり、E_{max} が最大べき指数である。算術演算の結果のべき指数が E_{max} より大きければ、べき指数オーバーフロー表示が生成されるべきである。同様に、E_{min} より小さければ指数アンダーフロー表示を生成されるべきである。仮数はいつも正規化されたままであるので、どんなオーバーフローもべき指数から反映される。

べき指数オーバーフローで無限の特別な表現が、結果として使われる。他の可能性としては計算を止めたり、処理の中断をしたり、表現可能な最大数に結果を入れたりするのを含んでいる。もし指数アンダーフローが生じたら、0 の表現は結果として使われる。しかし、適切なべき指数アンダーフロー信号がやはり発生されなければならない。もし適切で中断がなければ 0 の結果を置くことが計算を続行することを可能にする。

浮動点数の完全な範囲は、次の図式で示されている。0 は F^+ か F^- のどちらの範囲にも含まれていないことに注意する。

[第 4 章 終]