Appendix　0と1で浮動小数点をどう表現し演算するのか

浮動小数点の計算方法と注意点

大貫 徹

● 浮動小数点データ・タイプ

ここではIEEE 754形式で，最も幅広く使われている浮動小数点フォーマットについて説明します．

IEEE 754形式には単精度と倍精度の二つの形式があります．単精度は32ビットのデータ幅に入るので，32ビット・マイコンの整数(int)型とメモリの占有サイズが同じです．また32ビットのレジスタに入るという特徴もあって，制御系ではよく利用されるデータ・タイプです．しかし，シュミレーションや応用物理系では演算中の精度を高く求められる例が多く，後述する倍精度浮動小数点形式が利用されています．

● 単精度浮動小数点形式

まず単精度浮動小数点形式を**図A**に示します．

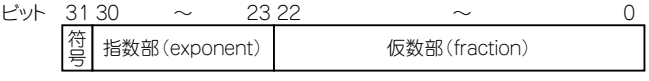
MSBは符号(sign)を表します．整数と同様にMSB=1のときに負の数であることを示しているので，マイコンはintと区別せずに符号の評価が行えます．指数部は浮動小数点の特徴ともいえる小数点の絶対位置を示しています．指数部は8ビットの整数で表されており，小数点の位置が仮数部のビット22の左側にあるとき，127(7Fh)の値を持ちます．小数点の位置がこれより右に位置するときは値がそのけた分だけ大きくなり，逆に左に位置するときは小さくなります．仮数部は実際の数値を2進数で表します．仮数部は23ビットのフィールドですが，実際にはビット23の位置に隠れた1が存在しているものとして取り扱います．

実際に値1.2345がどのように単精度フォーマットに収まるのかを例示してみましょう．2進数の性質に従って右側のけたは左側の半分の重みをもっているので，与えられた数値は次のように変換されます．

1.2345=1+0×0.5+0×0.25+1×0.125+1×0.0625+1×0.03125+1×0.015625+1×0.0078125+…以下省略(重みの係数部のみの回収結果が下記の2進数小数になる)

≒1.00111100000010000011001(これ以下のけたは切り捨て)

次に指数部ですが，最大のけたが小数点の左のけたなので，×2の冪(べき)項は0(仮数部の移動が不要)となり，IEEE 754の指数部オフセットである127(01111111)を加算した結果も127のままです．最後に，符号は正数なので0となり**図B**のようにパッキングされ，結果として3F9E0419h(00111111100111100000010000011001)となります．



図A　単精度浮動小数点形式(single precision floating-point format)

● 浮動小数点形式の性質

浮動小数点形式の性質に，数値が表現範囲内にある場合，2を掛けるあるいは2で割るという操作に対して，仮数部の値は変化しないというものがあります．例えば先の数値1.2345を2倍しても仮数部は変化せず，小数点の位置が変わるだけです．したがって変化は指数部の値が127から128に増えるだけなので，符号＋指数部＋仮数部の連結結果は次のようになります．

0+10000000+10.01111000000010000011001

=01000000000111100000010000011001

同様に1.2345を1/2にしても指数部が変化するのみであり，

0+01111110+0.10011110000010000011001

=00111111000111100000010000011001

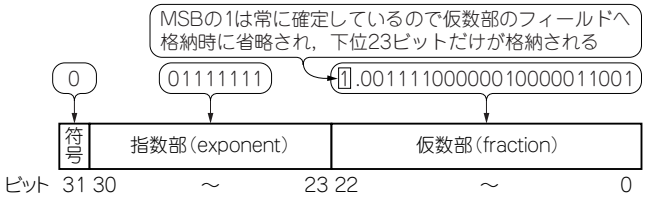
となります．小数点の位置が120以上も左右に移動可能なので，数値としての表現範囲が大きく拡大されたといえます(先頭のゼロは左詰となり常に1がビット23の位置にあるとする)．

ところで，2のベキ乗による乗除算の結果が常に仮数部に対して影響を与えないかというと，指数部が8ビットの整数で表現可能な範囲内であればという条件が付きます．では指数部は8ビットすべての表現範囲をとることが許されているかというと，IEEE 754の取り決めにより制限が設けられています．しかし指数部の極限値と仮数部の組み合わせを用いて特殊な数値表現を可能としています(**表A**)．

ここから**表A**の特殊表現を除いた最大と最小の数は次のようになります．最小は指数部が1で，仮数部が0すなわち23ビット目に隠れた1があるとする数で，$1 \times 2^{-126} \fallingdotseq 1.175494e-38$となります．

一方，最大は指数部が254で，仮数部がすべて1の数なので，2^{128}で約3.40282e+38になります．

このように最大の数を整数として表現しようとすると128ビットのけた数を必要とします．しかし実際は，仮数部としては24ビットの情報が最上位けた部にあるだけなので，下位けたの104ビットをすべて切り捨てているようなものです．よっ



図B　単精度でのパッキング

KEYWORD ―― 浮動小数点，IEEE754，単精度，倍精度，桁落ち

表A　IEEE 754での単精度特殊表現

符　号	指数部	仮数部	意　味
0	0	0	ゼロ（＋側のゼロへの極限）
1	0	0	ゼロ（－側のゼロへの極限）
-	0	≠0	非正規化数（アンダフロー）
0	255	0	＋∞
1	255	0	－∞
-	255	≧2^{22}	非数（quiet NaN）例外告知なし
-	255	2^{22}＞仮数部＞0	非数（signal NaN）例外告知あり

注：非数に対する演算結果はすべて無効

表B　IEEE 754での倍精度特殊表現

符　号	指数部	仮数部	意　味
0	0	0	ゼロ（＋側のゼロへの極限）
1	0	0	ゼロ（－側のゼロへの極限）
-	0	≠0	非正規化数（アンダフロー）
0	2047	0	＋∞
1	2047	0	－∞
-	2047	≧2^{51}	非数（quiet NaN）例外告知なし
-	2047	2^{51}＞仮数部＞0	非数（signal NaN）例外告知あり

注：非数に対する演算結果はすべて無効

て上記の最大数から2^{100}という大きな数を引いても，けた落ちにより0を引くため結果に全く影響を与えません．これが浮動小数点の計算で注意すべき点です．実質的な精度は単精度において24ビットであり，32ビット・マイコンのint型変数の精度31ビットより少なくなっています．

● 倍精度浮動小数点形式（Double Precision Floating-point Format）

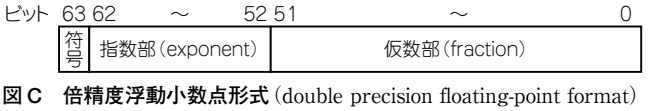
単精度では精度が不十分とされる応用物理系の計算処理においては，倍精度浮動小数点形式が利用されます．倍精度では64ビットの中に各コンポーネントが再配置され，精度が大きく拡張されています（**図C**）．

倍精度では指数部が11ビットに拡張され，仮数部は52ビット（実質53ビット）に拡張されました．これに伴い，極限値と特殊表現の取り扱いも**表B**のようになります．また，表現可能な最大と最小の値についても範囲が変わり，最小は1×2^{-1022}で2.2250738585072e－308，最大は1.79769313486232e+308へと拡大されます．

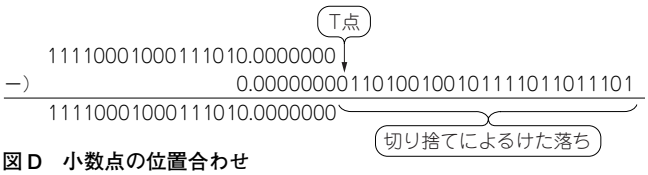
● 小数点を持つ数値計算の決まりごと

浮動小数点の仮数部は常に絶対値表現であり，補数は用いられません．これは先頭のゼロ・サプレスを実行し，最初に出てくる1のビットを単精度のビット23もしくは倍精度のビット52の位置に合わせるという正規化処理を行う必要があるためです．したがって，実際の加減算は二つの浮動小数点数値の符号同士の排他的論理和をとった後で，演算子が引き算ならさらに反転し，結果が1なら引き算，0なら加算を行います．

実際にアセンブラで浮動小数点の計算をすることは考えにくく，C言語で記述すると思われます．その場合は単純に数式どおりの記述で必要な演算が行われますが，加減算時のけた落ちには十分留意する必要があります．場合によっては演算の順序指定や，式に出てくる項の値の範囲についてあらかじめ分析し，計算が無効な処理とならぬよう検算しておく必要があります．どのようにしてけた落ちが発生するか，実際に単精度演算を例にとって説明しましょう．



図C　倍精度浮動小数点形式（double precision floating-point format）



図D　小数点の位置合わせ

ここでは123450－0.00321の計算を考えます．それぞれの単精度浮動小数点数値のバイナリは次のようになります．

(a) 123450 → 47F11D00h
　指数部+16　　仮数部1.11100010001110100000000
(b) 0.00321 → 3B525EDDh
　指数部-9　　仮数部1.10100100101111011011101

指数部の符号と値は127（7Fh）を中心とした場合の差分数値を示します．よって(a)の数値の本来の小数点位置は，仮数部に付けたものから右方向に16けた移動した位置にあります．

小数点の位置を合わせると次のようになります．

(a) 11110001000111010.0000000

一方，(b)の場合，小数点の位置を左方向に9けた動かす必要があります．このとき先頭にゼロを補填して小数点を配置し直すと，次のように表せます．

(b) 0.00000000110100100101111011011101

二つの小数点数を加減算する場合，お互いの小数点位置は同じ位置にそろえてから計算する必要があります．ではそろえてみましょう（**図D**）．

この引き算を試みた場合，数値aの最下位ビットよりも右側に引かれる数の実体があるのが見てとれます．もし加減算器の幅が24ビットしかないとすれば，bの数値はT点より下のけたは位置合わせに伴う右シフト時にすべて掃き出されて0を引いたことになり，まるまるすべてがけた落ちしてしまいます．

したがって，加減算においては元の数値が持っている有効けた数に注意し，有効けた数以下の数値との間で加減算をしないように計算式を考える必要があります．なぜなら，小さな数であっても小さな数同士であれば計算自体は有効な結果を出すからです．

おおぬき・とおる