≫ 文法の曖昧さを理解して確実性と再利用性を高める

マイコンC言語 転ばぬ先のつえ

第5回　浮動小数点数型②…高速・高効率に演算する方法

鹿取 祐二

リスト1　浮動小数点数型の演算で円の面積を求めるcircle関数
double型を使って円の面積を求めるプログラム．単純に見えるが，CPUはこのような演算が不得意なので，専用演算器「FPU」を使う

```
double circle(double r)
{
    return r * r * 3.141592;
}
```

C言語は，誕生から50年近く経つ今でも，分野を問わず，さまざまな場面で使われるプログラミング言語です．言語仕様（文法）は標準規格化されていますが，曖昧な部分が多く存在します．

本連載では，C言語の文法の曖昧な部分と，それにより起こる問題を解説します．再利用性と効率が高く，安全かつ安心して使えるソフトウェアが開発できるようになることを目指します．

第4回～第5回は，浮動小数点数型の文法に対する問題点や内容を解説します．〈編集部〉

浮動小数点数型を高速・高効率に演算する方法はマイコンごとに異なる

● 浮動小数点数型の演算はCPUにとっても複雑

浮動小数点数型の演算はとても複雑です．例えば加算する場合でも，仮数部だけを単純に加算することはできません．仮数部の各桁の重みは指数部の値によって変化するので，指数部が同じでなければ仮数部の単純な加算はできません．

このように，浮動小数点数型の演算は，整数型よりもはるかに複雑です．

このような複雑な浮動小数点数型の演算は，組み込み系のマイコンでも簡単に行えるのでしょうか．その答えは，FPU（Floating Point Unit）機能の有無によって変わります．CPU（Central Processing Unit）は，整数型やポインタ型の演算は得意ですが，浮動小数点数型の演算は不得意です．高速かつ効率良く浮動小数点数型を演算するには，FPUが必要です．

● 浮動小数点数型専用の演算器「FPU」が必須！

一般的にFPUは高価で，安価なマイコンには搭載されていません．ルネサス エレクトロニクス製マイコンの場合，H8とRL78には搭載されておらず，RXとSHの一部には搭載されています．

FPUの有無で浮動小数点数型の演算性能が大きく変化することを実際に確認してみます．**リスト1**に示すのは，半径を引数として円の面積を返すcircle関数です．

表1に各マイコンにおけるcircle関数実行時の使用メモリ量とクロック数を示します．RXとSHは，FPUあり（SH2A-FPU）/なし（SH2A）の両方の性能を示しています．

詳細は後述しますが，H8，SH2A，SH2A-FPUは条件をそろえるためにdouble型を単精度形式で扱うためのオプションを設定しました．結果を見ると，FPUの有無で性能が全然違うことが分かります．

● 演算の流れ

FPUの有無でこれほどまでの差が出るのは，なぜでしょうか．この理由は，コンパイル結果を見ると多少分かります．**リスト1**のソースコードをRL78（FPUなし）とRX（FPUあり）でコンパイルしたときの結果（アセンブリ・コード）を**リスト2**に示します．

表1　リスト1のcircle関数の実行結果
条件を合わせるため，H8とSH2Aには-double=float，SH2A-FPUには-fpu=singleを指定

項　目	RL78	H8/300H	RX（FPUなし）	RX（FPUあり）	SH2A	SH2A-FPU
FPUの有無	なし	なし	なし	あり	なし	あり
サイズ（circle関数単体）	25バイト	28バイト	17バイト	11バイト	26バイト	16バイト
サイズ（ランタイム・ライブラリ）	236バイト	386バイト	204バイト	−	284バイト	−
クロック・サイクル数（circle関数全体）	247クロック	822クロック	98クロック	4クロック	127クロック	5クロック