マイコンを正しく操縦するための作法

基礎から学ぶC言語講座

岡田 好一
Yoshikazu Okada

第9回 アセンブラとの連携とsin関数の作成

C言語は「高級アセンブラ」と呼ばれることがあり，つまりは機械寄りの記述ができる，とされています．それでもアセンブラは時に必要です．

アセンブラが要求される場面は，

(1)C言語が想定する計算機モデルと合わない処理が必要
(2)特に処理速度が要求される
(3)OSなどで特殊命令が必要

などです．

(1)は強引にC言語で組む場合もあるかと思いますが，ほかの高級言語風の記述と分けておくのが無難です．特に，機種依存部分は注釈などで要注意の記述であることを明記しておきます．(2)と(3)に関しては，R8Cの命令の特性を概観する必要があります．

今回は，アセンブラとC言語の連携について解説し，その応用として，R8Cの乗算器を生かした固定小数点sin関数の作成と，その使用例としてリサジュー図形表示プログラムを紹介します

効率的なプログラミングに役立つR8Cの特徴ある命令

パソコンやほかのマイコンの機械語を知っている方には，R8Cのほとんどの命令はごく普通に見えると思います．やや特異なのは，レジスタを経由しない，メモリ-メモリ間の転送や演算が可能な点です．

自分でアセンブリ言語で組むにしても，コンパイラの出力を見ても，プログラムの主要部分は転送・加算・比較・分岐命令で占められているはずです．そうした基本部分はR8Cでも注力されていて，素直に高速に実行されます．

さらに，R8Cには少数の特徴ある命令が存在します．以下では，それらについて説明します．

● 算術系

▶ mul/mulu：乗算

パソコンのCPUの乗算命令は珍しくありませんが，R8Cのような小規模のマイコンで高速の乗算が使えるのはありがたい点です．コンパイラは本命令を使います．

バレル・シフタは上位機種のみに装備されているので，移動桁数によっては乗算の方がシフトより高速になります．なお，シフト命令(sha/shl)には32ビット版が用意されています．

符号付き除算命令には，剰余の符号の付き方で2種類(div/divx)が用意されています．Cコンパイラはもちろんc言語標準のdiv命令を選びます．

▶ exts：符号拡張

符号付きの8ビット・データを16ビット・データ

Keyword 1 乗算器とバレル・シフタ

R8C/Tinyの魅力の一つが，きわめて高速の乗算器(multiplier)が使えることです．

まず素朴には，C言語でcharやintの掛け算に何のためらいもなくなります．わざわざシフト演算に変更する必要がありません．それどころか，場合によってはシフトの方が遅いこともありえます．ただし，R8Cのシフト命令は32ビット・レジスタも対象になるなど，別の利点があります．

バレル・シフタ(barrel shifter)と呼ばれる機構が内蔵されていれば，乗算器よりも高速に任意桁のシフトが可能です．マニュアルで演算時間が桁数に依存しないのですぐに判別できます．

しかし，R8Cのシリーズではかなり上位機種でないとバレル・シフタは採用されていません．

アセンブラともなると，乗算器のおかげで簡易な信号処理までが目標範囲になります．

実際，今回作成した固定小数点数のsin関数のクロック数を数えて，あまりに速いので少々感動してしまいました．

に代入したい場合などに使われます．比較・分岐しなくても，機械語に備え付けです．

▶ rmpa：積和(図9－1)

整数の配列の対応する要素を掛けて(符号付き)合計する命令です．配列をベクトルと見なすと，結果は内積に相当します．信号処理などで役立ちそうです．乗算器があるので高速です．

▶ abs：絶対値

あまりほかでは見ない命令です．比較・分岐するより高速です．素直に絶対値として使う以外に，折れ線グラフのような関数を作る際に役立ちそうです．

▶ adcf：キャリーの加算

キャリー付き加算で0を足したときと同じ働きですが，より高速です．

● ビット処理系

▶ bset/bclr/bnot/btst/bntst/band/bnand/bor/bnor/bxor/bnxor/bm*Cnd*：ビット演算

1ビット単位の演算命令です．ビット単位のアドレッシングが使われます．コンパイラは本命令群を使います．bm*Cnd*はbmnzなど，フラグ条件によって1ビットを転送する命令群です．

bset/bclr/bnotの以外では，C(キャリー)フラグが累算器として扱われます．Cフラグ自体に“1/0”をセットするのはfset/fclr命令です．Cフラグを反転させるにはbmnc命令が使えます．

▶ btstc/btsts：ビット・テストとセット

いわゆる「テスト・アンド・セット」命令です．メモリのテストと同メモリへの転送が不可分に実施されます．クリティカル・セクションのロックに使われる命令でもあり，排他・同期処理の基本命令です．

● その他

▶ rolc/rorc/rot：回転(図9－2)

rolcとrorcはレジスタやメモリの1ビットのシフトで，Cフラグを含めて回転する感じで移動します．続けて使うと，メモリ上の多倍長のデータが1ビットぶんシフトします．

rotはレジスタやメモリ(8/16ビット)内で任意のビットぶん回転します．

▶ smovf/smovb/sstr：ストリング命令

メモリの連続する領域をまとめてコピーしたり，8/16ビットの特定データで埋めることができます．

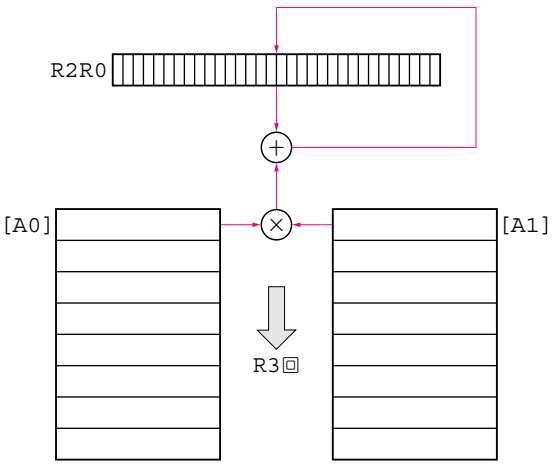

図9－1　積和演算命令rmpa(16ビット版)の動作
配列をベクトルと見なすと，結果は内積に相当．R3は演算回数

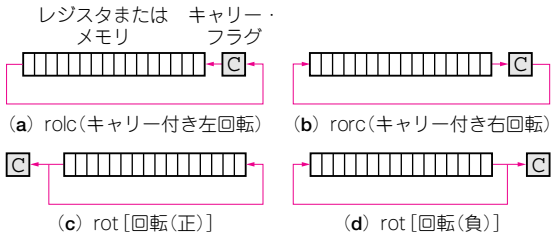

図9－2　回転命令rolc/rorc/rot(16ビット版)の動作
rolcとrorcは続けて使うとメモリ上の多倍長データが1ビットぶんシフト．rotはレジスタやメモリ内で任意のビットぶん回転

Keyword 2　除算命令

多くの16ビットCPUには除算命令(divide)があります．R8C/1Bにも除算命令があり，高速です．

しかし，乗算が非常に高速なので，それに比べれば時間はかかります．また，計算対象の数値によって演算時間が異なるのも特徴です．

比較的高速なので，C言語で気軽に割り算が使えるのは心強い点です．最内側のループでも，シフトを使って究明困難なバグに遭遇するよりは，割り算を直接使うと安心です．

R8Cには3種類の割り算命令があります．符号なし(divu)と2種の符号ありです．符号ありの場合，被除数と剰余の符号を合わせるdiv命令がC言語(C99)の規格と同じ動作です．

除数と剰余の符号を一致させるのがdivx命令です．被除数の正負で切り捨て・切り上げの方向が変わらないので，用途によっては役立つと思います．