ダウンロード・データあります

短期連載

より速く，より少ないメモリで動く効率的なプログラムを書くための

CPU最適化プログラミング

ご購入はこちら

第3回　SIMD命令でどのくらい最適化されたかを確認する方法

本橋 弘臣

本連載では，CPUを意識してプログラムを最適化することにより，効率的なプログラムを書くことを目指します．効率的なプログラムは，より速く，より少ないメモリで動作します．これは製品の完成度を高め，ユーザの体験を向上させることに直結します．

第3回はSIMD命令を使うことにより，どのくらい最適化されたかを確認する方法を紹介します．

（編集部）

表1　計測値を理論性能と比較

	実効性能	理論性能	利用率
演算性能［GFlops］	①	③	①÷③
メモリ・バンド幅［GB/s］	②	④	②÷④

プログラムの最適化度合いを確認する

● 計測値と理論性能を比較する

プログラムの最適化度合いを確認するための最も確実な方法は，出来上がったプログラムを実際に動作させてみて，処理性能を測定する方法です．

前回（2026年1月号）で説明した方法で①実効演算性能と②実効メモリ・バンド幅を計測し，その計測値を理論性能と比較し，利用率を見ることによって，プログラムの最適化がどの程度できているのかが分かります（**表1**）．

4×4行列乗算プログラムmat-mul.cのmat_mul()［**リスト1**(**a**)］，最適化版mat_mul_opt()［**リスト1**(**b**)］をCore i5-13400プロセッサ上で実行し

リスト1　4×4行列乗算プログラムmat-mul.c

```
019: #define N   4
省略
032: // 1次元配列を2次元配列的にアクセスするためのマクロ
033: #define mat_a_(x, y)    mat_a[(y) * N + (x)]
034: #define mat_b_(x, y)    mat_b[(y) * N + (x)]
035: #define mat_c_(x, y)    mat_c[(y) * N + (x)]

346: // 4x4行列の掛け算を計算する関数：ピュアCコード
347: void mat_mul(const float *mat_a, const float
                                  *mat_b, float *mat_c)
348: {
349:     int x, y, i;
350:     float sum;
351:
352:     // 行列 C = A × B の計算を行う
353:     for (y = 0; y < N; y++) {
354:         for (x = 0; x < N; x++) {
355:             sum = 0.0f;
356:             for (i = 0; i < N; i++) {
357:                 sum += mat_a_(i, y) *
                                        mat_b_(x, i);
                        // この行は64回実行する
358:             }
359:             mat_c_(x, y) = sum;
                        // この行は16回実行する
360:         }
361:     }
361: }
```

（a）mat_mul()

```
// 4x4 行列の掛け算を計算する関数：メモリ・アクセス改善版
void mat_mul_opt(const float *mat_a,
            const float *mat_b_trans, float *mat_c)
{
    int x, y;

    for (y = 0; y < N; y++) {
        for (x = 0; x < N; x++) {
            mat_c_(x, y) = dot_product(&mat_a_(
                        0, y), &mat_b_trans_(0, x));
        }
    }
}

// floatx4 ベクトルの内積を求める下請け関数：メモリ・アクセス改善版
INLINE float dot_product(const float *a,
                         const float *b)
{
    return a[0] * b[0] +
           a[1] * b[1] +
           a[2] * b[2] +
           a[3] * b[3];
}
```

（b）最適化版mat_mul_opt()

本連載筆者による特集記事
「SIMD/メモリ/AI機能…CPUに用意された拡張命令」
が本誌2025年11月号に掲載されています．