

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

新連載

第1回 SIMD 命令を使った行列演算の最適化

本橋 弘臣

リスト1 普通にC言語で4個のfloat値の乗算を書いた場合

```
void mul_float4(const float *a, const float *b,
               float *c)
{
    c[0] = a[0] * b[0];
    c[1] = a[1] * b[1];
    c[2] = a[2] * b[2];
    c[3] = a[3] * b[3];
}
```

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1回はSIMD命令を使った効率的なプログラムの書き方について解説します。(編集部)

SIMD命令を利用したプログラムの最適化

SIMD (Single Instruction Multiple Data) 命令^{注1}は、複数のデータに対して一気に計算を行うことが可能な命令です。SIMD命令はCPUに標準搭載されている汎用演算アクセラレータとも言えます。

SIMD命令を利用して最適化を行ったプログラムを開発するには、次の3通りの方法があります。

- ①普通にC言語でソースコードを書く
- ②Intrinsics関数を呼び出すCコードを書く
- ③アセンブリ言語でソースコードを書く

● 方法①…普通にC言語でソースコードを書く

ここでは最も簡単と思われる処理の例として、配列A[]とB[]にそれぞれ格納されている4個のfloat値に対して乗算を行い、4個の乗算結果を配列C[]に返す関数を考えてみます。①の方法でこの処理をC言語で書くと、リスト1のような関数になります。

● 方法②…Intrinsicsを呼び出すCコードを書く

IntrinsicsとはCコンパイラが提供する特別な関数のことで、1回の関数コールがSIMDのアセンブリ命令1個に対応しています。見た目は普通のCコードでありながら、アセンブリ言語でソースコードを記述するのと同程度の細かな粒度で、SIMD最適化版のプロ

注1：参考文献(1)には、インテルCPUで利用可能なSIMD命令の一覧表が掲載されていて、各SIMD命令の動作が分かりやすく解説されています。また各SIMD命令のニーモニック(アセンブリ命令)とIntrinsicsの対応関係も示されています。

表1
インテルCPU向け
Intrinsics一覧表：
float (FP32) 演算用

	Intrinsics	対応する アセンブラ命令	説明
SSE	_mm_loadu_ps()	MOVUPS	メモリから16バイト分のデータをSSEレジスタに読み込む
	_mm_add_ps()	ADDPS	SSEレジスタ上のfloat×4のデータ同士の加算を行う*1
	_mm_mul_ps()	MULPS	SSEレジスタ上のfloat×4のデータ同士の乗算を行う*2
	_mm_storeu_ps()	MOVUPS	SSEレジスタ上の16バイト分のデータをメモリに書き出す
	_mm_stream_ps()	MOVNTPS	SSEレジスタ上の16バイト分のデータを、キャッシュ・メモリをバイパスして直接メモリに書き込む
AVX	_mm256_loadu_ps()	VMOVUPS	メモリから32バイト分のデータをAVXレジスタに読み込む
	_mm256_add_ps()	VADDPS	AVXレジスタ上のfloat×8のデータ同士の加算を行う*1
	_mm256_mul_ps()	VMULPS	AVXレジスタ上のfloat×8のデータ同士の乗算を行う*2
	_mm256_storeu_ps()	VMOVUPS	AVXレジスタ上の32バイト分のデータをメモリに書き込む
	_mm256_stream_ps()	VMOVNTPS	AVXレジスタ上の32バイト分のデータを、キャッシュ・メモリをバイパスして直接メモリに書き込む

*1：ADDPS/VADDPS命令は(レジスタ)←(レジスタ)+(メモリ)の演算にも対応する。

*2：MULPS/VMULPS命令は(レジスタ)←(レジスタ)×(メモリ)の演算にも対応する。

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。

