SIMD 命令 でループを高速に

SIMD とは

- 1 命令で複数のデータに演算を適用できる
- 専用のレジスタを使うことが多い
  - 512-bit * 32 みたいな
- たとえば...の高速化に向いている
  - データの総和を求める
  - ベクトル，行列同士の演算

色々な SIMD 命令

- x86-64
  - SSE
  - AVX
- Arm
  - NEON
  - SVE

などなど

SIMD 命令を使ってみよう

- 今回は Intel, AMD の CPU で使える AVX2 命令を使う
- AVX2
  - 256-bit 幅のレジスタ
    - int なら一度に 8 個, double なら 4 個
  - 16 本のレジスタ
  - C 言語用の組み込み関数が用意されている
    - https://www.intel.com/content/www/us/en/docs/intrinsics-guide/index.html#avxnewtechs=AVX2

ベクトル同士の加算

```
void vecadd(float *C, float *A, float *B) {
  for (int i = 0; i < 1024; i++) {
    C[i] = A[i] + B[i];
  }
}
```

- ループ回数は 1024 回

AVX2 命令を適用

```
#include <immintrin.h>

void vecadd_avx2(float *C, float *A, float *B) {
  for (int i = 0; i < 1024; i += 8) {
    __m256 A_vec = _mm256_load_ps(&A[i]);
    __m256 B_vec = _mm256_load_ps(&B[i]);
    __m256 C_vec = _mm256_add_ps(A_vec, B_vec);
    _mm256_store_ps(&C[i], C_vec);
  }
}
```

- ループ回数が 1024 / 8 = 128 回に!
  - 8 倍高速化できる...?

計測

計測用マクロを用意

```
#include <stdio.h>
#include <time.h>

#define measure_cputime(tag, proc)                                  \
  {                                                                 \
    clock_t start, end;                                             \
    start = clock();                                                \
    {proc};                                                         \
    end = clock();                                                  \
    printf(tag ": %fs\n", ((double)(end - start)) / CLOCKS_PER_SEC);  \
  }
```

CPU 時間(≒ 実時間)だと思って計測する

コンパイル，実行

Apple M1 では AVX2 使えない，TUT の HPC クラスタ(のログインノード)を使う

```
module load gcc-7.3.1
gcc -O0 -mavx2 -std=c11 simd.c
./a.out
vecadd: 2.930000s
vecadd_avx2: 0.770000s
OK
```

- 2.93 / 0.77 ≒ 3.8 倍!
    - メモリから AVX2 レジスタへのロード，ストアがネックになっているかも

もっと SIMD 命令 を有効活用してみよう

次回につづく