一、背景介绍

SIMD（Single Instruction Multiple Data）指令集，指单指令多数据流技术，可用一组指令对多组数据通进行并行操作。详细参考 https://blog.csdn.net/qq_32916805/article/details/117637192。

二、常见数据类型与函数命名规则

1、 常用数据类型命名规则

- 每一种类型，从 1 个下划线开头，接一个 m，然后是向量的位长度
- 如果向量类型是以 d 结束的，那么向量里面是 double 类型的数字。如果没有后缀，就代表向量只包含 float 类型的数字
- 整形的向量可以包含各种类型的整形数，例如 char,short,unsigned long。也就是说，__m256i 可以包含 32 个 char，16 个 short 类型，8 个 int 类型，4 个 long 类型。这些整形数可以是有符号类型也可以是无符号类型

数据类型	描述
__m128	包含4个float类型数字的向量
__m128d	包含2个double类型数字的向量
__m128i	包含若干个整型数字的向量
__m256	包含8个float类型数字的向量
__m256d	包含4个double类型数字的向量
__m256i	包含若干个整型数字的向量

2、 常用函数命名规则

函数命名：`_mm<bit_width>_<name>_<data_type>`

`<bit_width>` 表明了向量的位长度，即操作对象的数据类型大小，对于 128 位的向量，这个参数为空，对于 256 位的向量，这个参数为 25

`<name>`描述了内联函数的算术操作。一般由两部分组成

第一部分是表示指令的作用，比如加法 add 等;
第二部分是可选的修饰符，表示一些特殊的作用，比如从内存对齐，逆序加载等

`<data_type>` 表明了操作的粒度，具体情形见下表

<data_type>标识		数据类型
epi8/epi16/epi32		有符号的8,16,32位整数
epu8/epu16/epu32		无符号的8,16,32位整数
si128/si256		未指定的128,256位向量
ps		包装型单精度浮点数
ss	scalar single precision floating point data	数量型单精度浮点数
pd	pached double precision floating point data	包装型双精度浮点数
sd		数量型双精度浮点数

可选的修饰符	示例	描述
u	loadu	Unaligned memory: 对内存未对齐的数据进行操作
s	subs/adds	Saturate: 饱和计算将考虑内存能够存储的最小/最大值。非饱和计算略内存问题。即计算的上溢和下溢
h	hsub/hadd	Horizontally: 在水平方向上做加减法
hi/lo	mulhi	高/低位
r	setr	Reverse order: 逆序初始化向量
fm	fmadd	Fused-Multiply-Add(FMA)运算，单一指令进行三元运算

三、SIMD 工作流程

加载：常用 set 或 load 函数，数据从内存进入 CPU 寄存器;
处理：数据处理;

存回：常用 store 函数，数据从 CPU 寄存器进入内存

四、SIMD 使用方法

1、 需要包含头文件#include <x86intrin.h>

2、编译 SIMD 函数库需要添加编译选项-msse2 -mssse3 -msse4.1 -msse4.2 -mavx

其中-msse2 -mssse3 -msse4.1 -msse4.2 用来支持 sse 指令集，寄存器为 128 位，-mavx 用来支持 avx 指令集，寄存器为 256 位

3、 -march=native 选项告诉编译器为当前的 CPU 架构和指令集生成优化的代码

Makefile 使用时可以如下

```
GCC = g++
CFLAGS = -O2 -std=c++14
SSEFLAGS = -msse2 -mssse3 -msse4.1 -msse4.2 -mavx -march=native
$(GCC) $(CFLAGS) $(SSEFLAGS) -o throughput.out throughput.cpp
```