Рекомендуется использовать инструкции AVX на процессорах архитектуры x86(-64); использование AVX2, AVX-512F и более новых наборов приветствуется. Пример для компилятора G++: чтобы подключить набор AVX2, укажите при компиляции ключ *-mavx2*. При этом убедитесь, что ваш процессор поддерживает эти инструкции, иначе программа, скорее всего, не запустится.

1. Реализуйте функцию *func*, возвращающую значение типа *double* и принимающую три аргумента:
- «сырой» указатель на неизменяемый массив *a* 64-битных чисел с плавающей точкой (*const double** a);
- «сырой» указатель на неизменяемый массив *b* того же типа (*const double** b);
- количество элементов в обоих массивах *n* (*int* n).

Функция должна работать корректно при **любом** *n* от 0 до (как минимум) 100 миллионов. Варианты:

- Залевский Александр:
```
(a[0] / b[0]) + (a[1] / b[1]) + ... + (a[n-1] / b[n-1])
```

- Латышев Артём:
```
min(2*a[0],3*b[0]) + min(2*a[1],3*b[1]) + ... + min(2*a[n-1],3*b[n-1])
```

- Юхимчук Александр:
```
max(2*a[0] + 3*b[0], 2*a[1] + 3*b[1], ..., 2*a[n-1] + 3*b[n-1])
```

2. Замерьте время работы функции *func*, скомпилировав программу с флагом -O0 (без оптимизаций). Прогоните функцию 110 раз на двух массивах размера *n* по крайней мере 100 миллионов: первые 10 прогонов – «холостые», остальные 100 необходимо измерить. Выведите среднее время работы функции в микросекундах. Для замеров рекомендуется библиотека <*chrono*>.

Массивы инициализируйте с помощью генератора псевдослучайных чисел (см. <*random*>). Перед каждым запуском *func* обновляйте содержимое массивов: например, перемешивайте элементы или генерируйте новые значения. Постарайтесь подобрать данные так, чтобы результат работы функции наглядно отличался от запуска к запуску.

3. Замерьте время работы функции *func*, скомпилировав программу с флагом -O3, аналогично предыдущему пункту. Сравните время работы.

4. Дизассемблируйте исполняемый файл программы, скомпилированной с флагом -O3. Это можно сделать как с помощью утилиты локально (например, objdump), так и с помощью онлайн-сервисов (например, Godbolt). Обратите внимание на инструкции в дампе. Произошла ли автоматическая векторизация? Найдите основные инструкции, выполняющие манипуляции с данными.

5. Проделайте пункт 4, только вместо флага -O3 укажите -Ofast. Объясните разницу с предыдущим пунктом.

6. Выполните ручную векторизацию: перепишите функцию *func* с использованием Intel Intrinsics. Функция по-прежнему должна работать корректно для всех допустимых *n*.

Руководство по расширенным инструкциям см. по ссылке ниже: там указано, какие заголовки подключить, синтаксис функций и т. д. Подсказка: можно подсмотреть в листинг из п. 5, но необязательно делать всё в точности так, как там написано.

7. Замерьте время работы вручную векторизованной *func*, скомпилировав программу с флагом -O0, аналогично пункту 2. Сравните результат с автоматической векторизацией и с запуском без оптимизаций. Объясните разницу.

8. Кэширование результатов. Модифицируйте вручную векторизованную функцию *func*: каждые *k* итераций кэшируйте очередную порцию из 8*k* байтов в массивах *a* и *b*. Для этого используйте функцию *_mm_prefetch*, но будьте осторожны, чтобы случайно не вылезти за пределы массива при кэшировании.
Сделайте запуски при *k* = 64, 1024, 32768, 1048576 и замерьте время. Сравните результаты. При каких *k* функция *func* работает быстрее? Попробуйте объяснить почему.

Полезные ссылки:
https://godbolt.org/
https://www.laruence.com/sse/