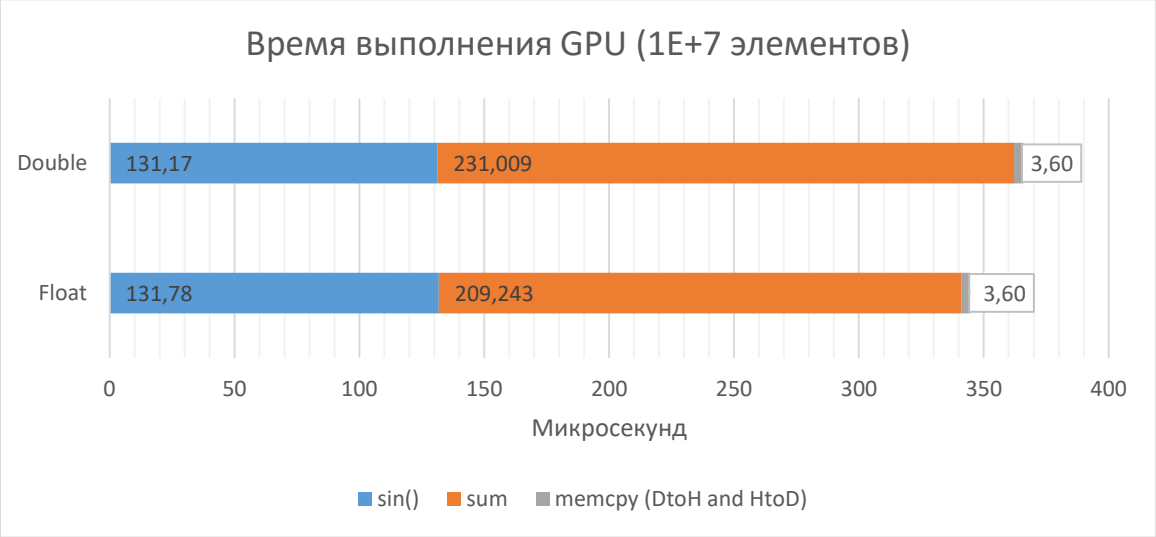
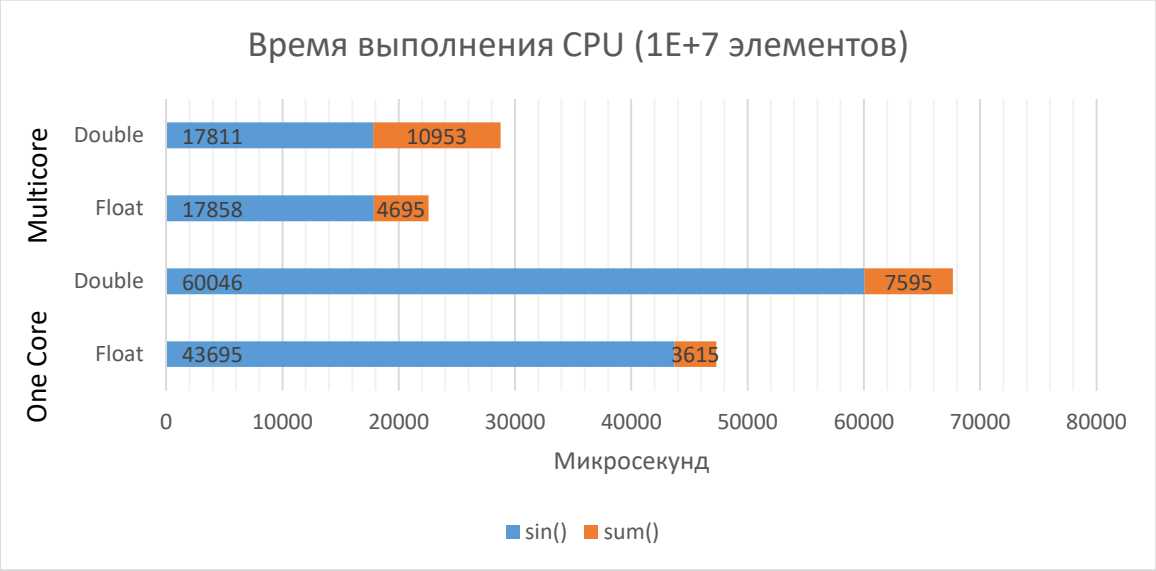
Сумма синусов

Компилятор для создания кода для видеокарты: pgc++ с флагами –fast –acc
Компилятор для создания кода для процессора: pgc++ с флагами –fast –Mconcur=allcores (Для многопоточного теста)

Исполняется на	Тип данных	Общее время выполнения	Время выполнения цикла на заполнения sin()	Время выполнения цикла на суммирование (+reduction)	Результат работы
GPU	Float	482824 us	131.78 us	209.243 us	-0.02853935546875000
	Double	479460 us	131.17 us	231.009 us	-0.00000000000321165317
CPU (One-core)	Float	47311 us	43695 us	3615 us	-0.04301386326551437378
	Double	67642 us	60046 us	7595 us	-0.00000000003578329730
CPU (Multi-core)	Float	22554 us	17858 us	4695 us	-0.04301386326551437378
	Double	28766 us	17811 us	10953 us	-0.00000000003578329730





```cpp
#include <iostream>
#include <cmath>
#include <chrono>
#include <iomanip>
#define arraySize 10000000
#ifdef Double
using floatingType = double;
#else
using floatingType = float;
#endif
int main(int argc, char const *argv[])
{
    double pi = acos(-1);
    floatingType sum = 0;
    floatingType* array = new floatingType[arraySize];

    auto allProgramStart = std::chrono::high_resolution_clock::now();

    #pragma acc enter data create(array[0:arraySize], sum) copyin(pi)
#ifdef CPU
    auto allCycles = std::chrono::high_resolution_clock::now();
    auto firstCycle = std::chrono::high_resolution_clock::now();
#endif

    #pragma acc parallel loop present(array[0:arraySize],sum)
    for(int i = 0; i < arraySize; i++){
        array[i] = sin(2 * pi*i/arraySize);
    }

#ifdef CPU
    auto firstCycleElapsed = std::chrono::high_resolution_clock::now() - firstCycle;
    auto secondCycle = std::chrono::high_resolution_clock::now();
#endif

    #pragma acc parallel loop present(array[0:arraySize],sum) reduction(+:sum)
    for(int i = 0; i < arraySize; i++){
        sum += array[i];
    }
#ifdef CPU
    auto allCyclesElapsed = std::chrono::high_resolution_clock::now() - allCycles;
    auto secondCycleElapsed = std::chrono::high_resolution_clock::now() - secondCycle;
#endif
    #pragma acc exit data copyout(sum) delete(array[0:arraySize], pi)

    auto allProgramElapsed = std::chrono::high_resolution_clock::now() - allProgramStart;
    long long programMicro =
std::chrono::duration_cast<std::chrono::microseconds>(allProgramElapsed).count();
    std::cout << std::fixed;
#ifdef CPU
    long long allCyclesMicro =
std::chrono::duration_cast<std::chrono::microseconds>(allCyclesElapsed).count();
    long long firstCyclesMicro =
std::chrono::duration_cast<std::chrono::microseconds>(firstCycleElapsed).count();
    long long secondCyclesMicro =
std::chrono::duration_cast<std::chrono::microseconds>(secondCycleElapsed).count();
    std::cout << "Two cycles " << allCyclesMicro << " us" << std::endl;
    std::cout << "First cycle " << firstCyclesMicro << " us" << std::endl;
    std::cout << "Second cycle " << secondCyclesMicro << " us" << std::endl;
#endif
    std::cout << "Programm " << programMicro << " us" << std::endl;
    std::cout << std::setprecision(20) << sum << std::endl;
    delete[] array;
    return 0;
}
```

В случае если нам необходимо провести работу над небольшим количеством данных, быстрее будет производить запуск на CPU (Необходимое время подготовки меньше). Погрешность в обоих случая минимальна и для double абсолютно незначительна. Но при вычислении на GPU она меньше. Оптимальным решением данной задачи для массива на 1.E+7 элементов будет решение на CPU (меньше общее время программы).