МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра вычислительной математики и программирования

**спецкурс «Параллельные и распределенные вычисления»**

**ОТЧЕТ**

**Лабораторная работа № 1**

**«Вычисление функции косинуса»**

Выполнил: Сотников И. Д.

Группа: М8О-207М-20

Преподаватель: Семенов С. А.

Москва, 2021

Содержание

[1. Постановка задачи 2](#_Toc52570380)

[2. Описание решения 2](#_Toc52570381)

[3. Аппаратное обеспечение и ПО 2](#_Toc52570382)

[4. Основные моменты кода 3](#_Toc52570383)

[5. Результат работы программы 5](#_Toc52570384)

[6. Сравнение скорости выполнения на CPU и GPU 6](#_Toc52570385)

[7. Выводы 6](#_Toc52570386)

[8. Приложения 7](#_Toc52570387)

# 1. Постановка задачи

Вычислить функцию косинуса.

# 2. Описание решения

В ходе решения будет использоваться функция из CUDA Math API:

\_\_device\_\_​ float cosf ( float  x );

Функция возвращает значение косинуса аргумента. Значение аргумента должно быть выражено в радианах.

# 3. Аппаратное обеспечение и ПО

Процессор: AMD Ryzen 5 3600 (6 core, 12 thread, 3.6 GHz);

RAM: 16 GB;

Графический процессор: Nvidia GeForce GTX 460 SE (

Total amount of global memory: 1024 MB

(6) Multiprocessors, (48) CUDA Cores/MP: 288 CUDA Cores

GPU Max Clock rate: 1460 MHz

Memory Clock rate: 1700 MHz

Memory Bus Width: 256-bit);

CUDA Driver Version / Runtime Version: 8.0 / 8.0;

CUDA Capability Major/Minor version number: 2.1.

# 4. Основные моменты кода

Эмулируем входные данные с помощью случайного распределения от 0 до 1:

|  |
| --- |
| **Функция случайной величины** |
| float randFloat(float min, float max) {  return (max - min) \* ((((float)rand()) / (float)RAND\_MAX)) + min;  } |

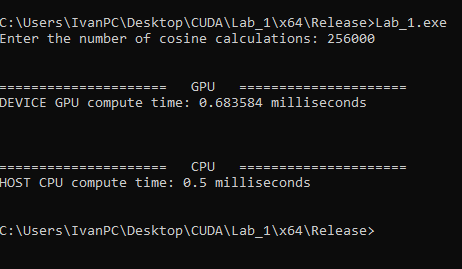
Чтобы оценить время работы в зависимости от размера входных данных, будет вычисляться вектор размерности *N*. Зададим число блоков равным 8, тогда число нитей в каждом блоке будет *N* / 8.

|  |
| --- |
| **Ядро, выполняющееся на device:** |
| \_\_global\_\_  void cos\_gpu(float \*arg, float \*res) {  res[threadIdx.x + blockIdx.x \* blockDim.x] = cosf(arg[threadIdx.x + blockIdx.x \* blockDim.x]);  } |

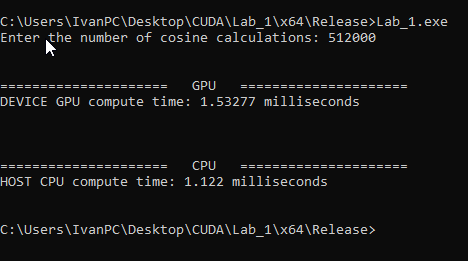
|  |
| --- |
| **Функция расчета косинуса на GPU (BLOCK\_NUM = 8):** |
| void gpu\_compute(int N) {  float \*host\_args = new float[N];  float \*host\_res = new float[N];  for (auto i = 0; i < N; i++)  host\_args[i] = randFloat(0, 1);  float \*device\_args, \*device\_res;  const int size = N \* sizeof(float);  // начало отсчета времени  cudaEvent\_t start, stop;  float gpuTime = 0.0f;  cudaEventCreate(&start);  cudaEventCreate(&stop);  cudaEventRecord(start, 0);  // выделение памяти под переменные в device  cudaMalloc(&device\_args, size);  cudaMalloc(&device\_res, size);  // копироваание переменной из host в device  cudaMemcpy(device\_args, host\_args, size, cudaMemcpyHostToDevice);  // вычисление функции  cos\_gpu <<<BLOCK\_NUM, N / BLOCK\_NUM >>> (device\_args, device\_res);  // дожидаемся выполнения  cudaThreadSynchronize();  // копироваание переменной из device в host  cudaMemcpy(host\_res, device\_res, size, cudaMemcpyDeviceToHost);  // конец отсчета времени  cudaEventRecord(stop, 0);  cudaEventSynchronize(stop);  cudaEventElapsedTime(&gpuTime, start, stop);  // очистка памяти  cudaFree(device\_args);  cudaFree(device\_res);  delete[] host\_args;  delete[] host\_res;  std::cout << "\n\n===================== GPU =====================\n";  std::cout << "DEVICE GPU compute time: " << gpuTime << " milliseconds\n\n";  } |

# 5. Результат работы программы

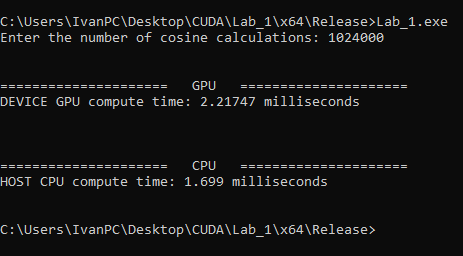
При *N* = 256000:



При *N* = 512000:



При *N* = 1024000:

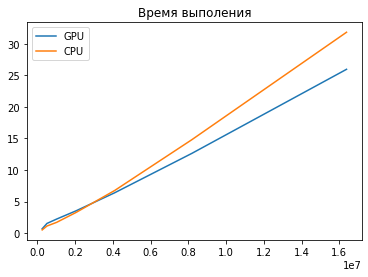


# 6. Сравнение скорости выполнения на CPU и GPU

При запуске программы с различными значениями *N* видно, что с повышением *N* вычисления на видеокарте становятся быстрее, но не на много, так используется сильно устаревшая видеокарта с относительно современным процессором.

Время выполнения программы при различных значениях *N*:

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **GPU**  **время выполнения, мс** | **CPU**  **время выполнения, мс** | **tCPU/tGPU** |
| 256000 | 0.68 | 0.5 | 0.735 |
| 512000 | 1.53 | 1.12 | 0.73 |
| 1024000 | 2.22 | 1.7 | 0.765 |
| 2048000 | 3.52 | 3.26 | 0.926 |
| 4096000 | 6.38 | 6.74 | 1,056 |
| 8192000 | 12.63 | 14.8 | 1.172 |
| 16384000 | 25.94 | 31.81 | 1.226 |



**Рис.** График зависимости времени выполнения программы от размера входных данных *N*.

# 7. Выводы

В Лабораторной работе №1 написаны программы подсчета функции косинуса от вектора значений аргументов, выполняющиеся на CPU и GPU соответственно. Был проведен анализ времени работы и сравнение двух реализаций подсчета функции на CPU и GPU.

# 8. Приложения

Проект Visual Studio: https://github.com/L0F1/CUDA\_labs