

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ  
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра вычислительной математики и программирования

**спецкурс «Параллельные и распределенные вычисления»**

**ОТЧЕТ**

**Лабораторная работа № 1**  
**«Освоение программного обеспечения среды программирования**  
**NVIDIA»**

Выполнил: Миронов С.В.  
Группа: М8О-103М-20  
Преподаватель: Семенов С. А.

## Содержание

|   |   |
|---|---|
| 1. Постановка задачи.....                           | 2 |
| 2. Описание решения .....                           | 2 |
| 3. Аппаратное обеспечение и ПО .....                | 2 |
| 4. Основные моменты кода .....                      | 2 |
| 5. Результат работы программы .....                 | 5 |
| 6. Сравнение скорости выполнения на CPU и GPU ..... | 5 |
| 7. Выводы.....                                      | 6 |

### 1. Постановка задачи

Вычислить функцию экспоненты

### 2. Описание решения

Разбиваем функцию экспоненты в бесконечный ряд:  $1 + x/1! + x^2/2! + \dots$ , и считаем каждый член этого ряда на отдельном вычислителе, в зависимости от необходимой точности меняя количество членов ряда

### 3. Аппаратное обеспечение и ПО

Программа должна быть скомпилирована с опцией Release и запускаться на Windows 7,10

CUDA Toolkit 7 и выше.

Программа должна быть скомпилирована CUDA, OpenCl, OpenACC.4.

### Основные моменты кода

Функция main:

```

int main()
{
    cudaError_t cudaStatus;
    int arraySize = 8;
    double x[1] = { 1 };
    //long double c[arraySize] = { 0 };

    std::cout << "Enter X: ";
    std::cin >> x[0];

    for (int j = arraySize; j < 2500; j *= 2) {
        calculateE_gpu(x, j);
        printf("\n");
        calculateE_cpu(x[0], j);
        printf("\n");
    }

    // cudaDeviceReset must be called before exiting in order for profiling and
    // tracing tools such as Nsight and Visual Profiler to show complete traces.
    cudaStatus = cudaDeviceReset();
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaDeviceReset failed!");
        return 1;
    }

    return 0;
}

```

Функция для вызова вычислений на gpu

```

void calculateE_gpu(double *x, int arraySize) {
    long double *c = new long double[arraySize];
    auto begin = std::chrono::steady_clock::now();

    cudaError_t cudaStatus = addWithCuda(c, arraySize, x);

    auto end = std::chrono::steady_clock::now();
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "addWithCuda failed!");
    }

    long double e = 0;
    for (int i = 0; i < arraySize; i++) {
        e += c[i];
    }
    printf("With GPU(%i components)e^%lf = %.16lf \n", arraySize, x[0], e);
    auto elapsed_ms = std::chrono::duration_cast<std::chrono::nanoseconds>(end - begin);
    printf("Time= %i nanoseconds", elapsed_ms.count());
}

```

Функция, выполняющаяся на gpu

```

__global__ void addKernel(long double *c, const double *x)
{
    int i = threadIdx.x;
    c[i] = power(x[0], i) / fact(i);
}

```

## 5. Результат работы программы

```
Консоль отладки Microsoft Visual Studio
Enter X: 1
With GPU(8 components)e^1.000000 = 2.7182539682539684
Time= 1205836700 nanoseconds
With CPU(8 components)e^1.000000 = 2.7182539682539684
Time= 800 nanoseconds
With GPU(16 components)e^1.000000 = 2.7182818284589949
Time= 767500 nanoseconds
With CPU(16 components)e^1.000000 = 2.7182818284589949
Time= 1400 nanoseconds
With GPU(32 components)e^1.000000 = 2.7182818284590455
Time= 658100 nanoseconds
With CPU(32 components)e^1.000000 = 2.7182818284590455
Time= 4000 nanoseconds
With GPU(64 components)e^1.000000 = 2.7182818284590455
Time= 619400 nanoseconds
With CPU(64 components)e^1.000000 = 2.7182818284590455
Time= 13100 nanoseconds
With GPU(128 components)e^1.000000 = 2.7182818284590455
Time= 671200 nanoseconds
With CPU(128 components)e^1.000000 = 2.7182818284590455
Time= 49200 nanoseconds
With GPU(256 components)e^1.000000 = 2.7182818284590455
Time= 734200 nanoseconds
With CPU(256 components)e^1.000000 = 2.7182818284590455
Time= 193000 nanoseconds
With GPU(512 components)e^1.000000 = 2.7182818284590455
Time= 899400 nanoseconds
With CPU(512 components)e^1.000000 = 2.7182818284590455
Time= 766900 nanoseconds
With GPU(1024 components)e^1.000000 = 2.7182818284590455
Time= 1434700 nanoseconds
With CPU(1024 components)e^1.000000 = 2.7182818284590455
Time= 3074200 nanoseconds
```

## 6. Сравнение скорости выполнения на CPU и GPU

При запуске программы с различными значениями  $N$  видно, что вычисления на видеокарте производились быстрее, чем на процессоре компьютера, при  $N > 512$ .

Время выполнения программы при различных значениях  $N$  (с учетом выделения памяти на GPU):

| <b>N</b> | <b>GPU<br/>время выполнения,<br/>нс</b> | <b>CPU<br/>время выполнения,<br/>нс</b> | <b><math>t_{CPU}/t_{GPU}</math></b> |
|----------|---|---|-------------------------------------|
| 16       | 767500                                  | 1400                                    | 0,0018241                           |
| 32       | 658100                                  | 4000                                    | 0,0060781                           |
| 64       | 619400                                  | 13100                                   | 0,0211495                           |
| 128      | 671200                                  | 49200                                   | 0,07330155                          |
| 256      | 734200                                  | 193000                                  | 0,26287115                          |
| 512      | 899400                                  | 766900                                  | 0,85267956                          |
| 1024     | 1434700                                 | 3074200                                 | 2,14274761                          |



**Рис.** График зависимости времени выполнения программы от количества членов  $N$ .

## 7. Выводы

В Лабораторной работе №1 проведен анализ работы различных программ по решению задачи для нахождения функции  $e^x$ , и выяснен предел эффективности CPU по сравнению с GPU для данной задачи