Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра "Электронные вычислительные машины"

**Отчет**

по лабораторной работе №1

«Знакомство с OPENCL»

Выполнили:

студенты гр. 045

Прохин А. И.

Фролов Д. Е.

Проверили:

доц. Муратов Е. Р.

доц. Устюков Д. И.

Рязань, 2023 г.

**Цель работы:** Получение практических навыков программирования на OpenCL.

**Практическая часть.**

**Задания 1 – 2:** Обработать буфер входных данных по формуле O=I+K, где I - символ (типа char) во входном буфере, О - символ (типа char) в выходном буфере, K - значение добавляемое к коду символа. Набрать программу по пунктам методички, вывести результат конвертирования на экране с применением GPU.

**Код:**

#include <CL/cl.h>

#include <iostream>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <string>

#include <fstream>

using namespace std;

int convertToString(const char\* filename, std::string& s)

{

size\_t size;

char\* str;

std::fstream f(filename, (std::fstream::in | std::fstream::binary));

if (f.is\_open())

{

size\_t fileSize;

f.seekg(0, std::fstream::end);

size = fileSize = (size\_t)f.tellg();

f.seekg(0, std::fstream::beg);

str = new char[size + 1];

if (!str)

{

f.close();

return 0;

}

f.read(str, fileSize);

f.close();

str[size] = '\0';

s = str;

delete[] str;

return 0;

}

return -1;

}

void Init()

{

cl\_uint numPlatforms;

cl\_platform\_id platform = NULL;

clGetPlatformIDs(0, NULL, &numPlatforms);

if (numPlatforms > 0)

{

cl\_platform\_id\* platforms = (cl\_platform\_id\*)malloc(numPlatforms \* sizeof(cl\_platform\_id));

clGetPlatformIDs(numPlatforms, platforms, NULL);

platform = platforms[0];

free(platforms);

}

else

{

cout << "No OpenCL platforms found." << endl;

return;

}

cl\_uint numDevices = 0;

clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_GPU, 0, NULL, &numDevices);

cl\_device\_id\* devices = (cl\_device\_id\*)malloc(numDevices \* sizeof(cl\_device\_id));

clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_GPU, numDevices, devices, NULL);

cl\_context context = clCreateContext(NULL, 1, devices, NULL, NULL, NULL);

cl\_command\_queue commandQueue = clCreateCommandQueue(context, devices[0], 0, NULL);

const char\* filename = "HelloWorld.cl";

string sourceStr;

int status = convertToString(filename, sourceStr);

if (status != 0)

{

cout << "Failed to read kernel file." << endl;

return;

}

const char\* source = sourceStr.c\_str();

size\_t sourceSize[] = { strlen(source) };

cl\_program program = clCreateProgramWithSource(context, 1, &source, sourceSize, NULL);

clBuildProgram(program, 1, devices, NULL, NULL, NULL);

const char\* input = "Hello, World!";

size\_t strlength = strlen(input);

char\* output = (char\*)malloc(strlength + 1);

cl\_mem inputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, (strlength + 1) \* sizeof(char), (void\*)input, NULL);

cl\_mem outputBuffer = clCreateBuffer(context, CL\_MEM\_WRITE\_ONLY, (strlength + 1) \* sizeof(char), NULL, NULL);

cl\_kernel kernel = clCreateKernel(program, "helloworld", NULL);

int K = 1; // Произвольное значение, которое будет добавлено к коду символа

clSetKernelArg(kernel, 2, sizeof(int), (void\*)&K); // Установить аргумент K

clSetKernelArg(kernel, 0, sizeof(cl\_mem), (void\*)&inputBuffer);

clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&outputBuffer);

size\_t global\_work\_size[1] = { strlength };

clEnqueueNDRangeKernel(commandQueue, kernel, 1, NULL, global\_work\_size, NULL, 0, NULL, NULL);

clEnqueueReadBuffer(commandQueue, outputBuffer, CL\_TRUE, 0, strlength \* sizeof(char), output, 0, NULL, NULL);

output[strlength] = '\0';

cout << "Input: " << input << endl;

cout << "Output: " << output << endl;

clReleaseKernel(kernel);

clReleaseProgram(program);

clReleaseMemObject(inputBuffer);

clReleaseMemObject(outputBuffer);

clReleaseCommandQueue(commandQueue);

clReleaseContext(context);

free(output);

free(devices);

}

int main()

{

Init();

return 0;

}

\_\_kernel void helloworld(\_\_global const char\* input, \_\_global char\* output, int K)

{

int i = get\_global\_id(0);

output[i] = input[i] + K;

}

**Результат работы программы:**

Input: Hello, World!

Output: Ifmmp-!Xpsme!

**Задание 4:** Создать по аналогии с предыдущими пунктами создать два входных буфера A и B с типом значений - float (не менее 10 значений в буфере). Инициализировать их значениями. С помощью OpenCL вычислить парное произведение A[i]\*B[i]. Вывести результат на мониторе (Рисунок 1).

**Код:**

#include <CL/cl.h>

#include <iostream>

#include <fstream>

#include <string>

void Init()

{

// Инициализация OpenCL

cl\_uint numPlatforms;

cl\_platform\_id platform = NULL;

clGetPlatformIDs(0, NULL, &numPlatforms);

if (numPlatforms > 0)

{

cl\_platform\_id\* platforms = (cl\_platform\_id\*)malloc(numPlatforms \* sizeof(cl\_platform\_id));

clGetPlatformIDs(numPlatforms, platforms, NULL);

platform = platforms[0];

free(platforms);

}

else

{

std::cout << "No OpenCL platforms found." << std::endl;

return;

}

cl\_uint numDevices = 0;

clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_GPU, 0, NULL, &numDevices);

cl\_device\_id\* devices = (cl\_device\_id\*)malloc(numDevices \* sizeof(cl\_device\_id));

clGetDeviceIDs(platform, CL\_DEVICE\_TYPE\_GPU, numDevices, devices, NULL);

cl\_context context = clCreateContext(NULL, 1, devices, NULL, NULL, NULL);

cl\_command\_queue commandQueue = clCreateCommandQueue(context, devices[0], 0, NULL);

// Чтение OpenCL ядра из файла

std::ifstream file("HelloWorld\_Kernel.cl");

std::string kernelSource(std::istreambuf\_iterator<char>(file), (std::istreambuf\_iterator<char>()));

const char\* kernelSourceCStr = kernelSource.c\_str();

cl\_program program = clCreateProgramWithSource(context, 1, &kernelSourceCStr, NULL, NULL);

clBuildProgram(program, 1, devices, NULL, NULL, NULL);

const int bufferSize = 10;

float A[bufferSize] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

float B[bufferSize] = { 9, 9, 9, 9, 9, 9, 9, 9, 9, 9 };

cl\_mem bufferA = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, bufferSize \* sizeof(float), A, NULL);

cl\_mem bufferB = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, bufferSize \* sizeof(float), B, NULL);

cl\_mem bufferC = clCreateBuffer(context, CL\_MEM\_WRITE\_ONLY, bufferSize \* sizeof(float), NULL, NULL);

cl\_kernel kernel = clCreateKernel(program, "multiply", NULL);

clSetKernelArg(kernel, 0, sizeof(cl\_mem), (void\*)&bufferA);

clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&bufferB);

clSetKernelArg(kernel, 2, sizeof(cl\_mem), (void\*)&bufferC);

size\_t globalWorkSize[1] = { bufferSize };

clEnqueueNDRangeKernel(commandQueue, kernel, 1, NULL, globalWorkSize, NULL, 0, NULL, NULL);

float\* result = new float[bufferSize];

clEnqueueReadBuffer(commandQueue, bufferC, CL\_TRUE, 0, bufferSize \* sizeof(float), result, 0, NULL, NULL);

for (int i = 0; i < bufferSize; ++i) {

std::cout << "A[" << i << "] \* B[" << i << "] = " << result[i] << std::endl;

}

delete[] result;

clReleaseMemObject(bufferA);

clReleaseMemObject(bufferB);

clReleaseMemObject(bufferC);

clReleaseKernel(kernel);

clReleaseProgram(program);

clReleaseCommandQueue(commandQueue);

clReleaseContext(context);

}

int main()

{

Init();

return 0;

}

\_\_kernel void multiply(\_\_global float\* A, \_\_global float\* B, \_\_global float\* C) {

int num = get\_global\_id(0);

C[num] = A[num] \* B[num];

}

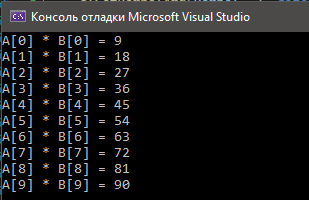


Рисунок 1 – Результат работы

**Задание 5:** Получить сведения об устройстве (Имя устройства, тип, количество вычислительных блоков и т.п.) (Рисунок 2).

**Код:**

#include <CL/cl.h>

#include <iostream>

int main() {

cl\_uint numPlatforms;

clGetPlatformIDs(0, NULL, &numPlatforms);

if (numPlatforms == 0) {

std::cout << "No OpenCL platforms found." << std::endl;

return 1;

}

cl\_platform\_id\* platforms = new cl\_platform\_id[numPlatforms];

clGetPlatformIDs(numPlatforms, platforms, NULL);

for (cl\_uint i = 0; i < numPlatforms; ++i) {

cl\_uint numDevices;

clGetDeviceIDs(platforms[i], CL\_DEVICE\_TYPE\_ALL, 0, NULL, &numDevices);

if (numDevices == 0) {

std::cout << "No OpenCL devices found for platform " << i << std::endl;

continue;

}

cl\_device\_id\* devices = new cl\_device\_id[numDevices];

clGetDeviceIDs(platforms[i], CL\_DEVICE\_TYPE\_ALL, numDevices, devices, NULL);

for (cl\_uint j = 0; j < numDevices; ++j) {

char deviceName[1024];

clGetDeviceInfo(devices[j], CL\_DEVICE\_NAME, sizeof(deviceName), deviceName, NULL);

cl\_device\_type deviceType;

clGetDeviceInfo(devices[j], CL\_DEVICE\_TYPE, sizeof(deviceType), &deviceType, NULL);

cl\_uint computeUnits;

clGetDeviceInfo(devices[j], CL\_DEVICE\_MAX\_COMPUTE\_UNITS, sizeof(computeUnits), &computeUnits, NULL);

std::cout << "Platform " << i << ", Device " << j << std::endl;

std::cout << "Device Name: " << deviceName << std::endl;

std::cout << "Device Type: ";

if (deviceType & CL\_DEVICE\_TYPE\_CPU) std::cout << "CPU ";

if (deviceType & CL\_DEVICE\_TYPE\_GPU) std::cout << "GPU ";

if (deviceType & CL\_DEVICE\_TYPE\_ACCELERATOR) std::cout << "Accelerator ";

std::cout << std::endl;

std::cout << "Number of Compute Units: " << computeUnits << std::endl;

std::cout << "--------------------------------------" << std::endl;

}

delete[] devices;

}

delete[] platforms;

return 0;

}

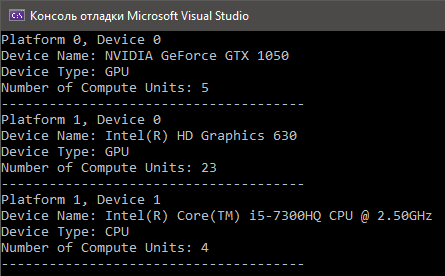


Рисунок 2 – Результат работы

**Вывод:** в результате выполнения лабораторной работы были получены практические навыки программирования на OpenCL.