Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический

университет имени В.Ф. Уткина»

Кафедра «ЭВМ»

Отчет о лабораторной работе №1

«Знакомство с OPENCL»

по дисциплине

«Специализированные ЭВМ»

Выполнили:

Студенты группы 045

Вашкулатов Н.А.

Анохин В.А.

Проверил:

доц. каф. ЭВМ Устюков Д.И.

доц. каф. ЭВМ Муратов Е.Р.

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

**Ход работы**

1. 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.
2. Набрать программу по пунктам методички, вывести результат конвертирования на экране с применением GPU (при наличии GPU с поддержкой OpenCL на лабораторном компьютере).

**Код программы:**

#include <CL/cl.h>

#include <string.h>

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

void HelloWOrld(const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue);

void Mul(const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue);

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 isSuccess(cl\_int status) {

#define CL\_SUCCESS 0

if (status != CL\_SUCCESS)

{

cout << "Ошибка: " << status << endl;

}

}

void Init() {

cl\_uint numPlatforms;

cl\_platform\_id platform = NULL;

cl\_int status = clGetPlatformIDs(0, NULL, &numPlatforms);

isSuccess(status);

cout << "NumPlatforms: " << numPlatforms << endl;

if (numPlatforms > 0)

{

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

status = clGetPlatformIDs(numPlatforms, platforms, NULL);

isSuccess(status);

platform = platforms[0];

free(platforms);

}

cl\_uint numDevices = 0;

cl\_device\_id\* devices;

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

isSuccess(status);

cout << "NumDevices: " << numDevices << endl;

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

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

isSuccess(status);

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

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

cout << endl << "Run programs" << endl << endl;

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

HelloWOrld(filename, context, devices, commandQueue);

const char\* filename1 = "Mul.cl";

Mul(filename1, context, devices, commandQueue);

status = clReleaseCommandQueue(commandQueue);

status = clReleaseContext(context);

if (devices != NULL)

{

free(devices);

devices = NULL;

}

}

void HelloWOrld( const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue)

{

cl\_int status;

string sourceStr;

status = convertToString(filename, sourceStr);

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

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

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

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

const char\* input = "GdkknVnqkc";

size\_t strlength = strlen(input);

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

const int k = 10;

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

cl\_mem inputBuffer2 = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, sizeof(int), (void\*)&k, NULL);

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

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

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

status = clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&inputBuffer2);

status = clSetKernelArg(kernel, 2, sizeof(cl\_mem), (void\*)&outputBuffer);

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

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

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

output[strlength] = '\0';

cout << output << endl;

status = clReleaseKernel(kernel);

status = clReleaseProgram(program);

status = clReleaseMemObject(inputBuffer);

status = clReleaseMemObject(outputBuffer);

if (output != NULL)

{

free(output);

output = NULL;

}

}

int main()

{

Init();

return 0;

}

**Код программы OpenCL:**

\_\_kernel void helloworld( global char\* in,global int\* k, global char\* out)

{

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

out[num] = in[num] + \*k;

}

**Окно вывода: **

1. Выполнить программу c применением CPU (меняется константа с CL\_DEVICE\_TYPE\_GPU на CL\_DEVICE\_TYPE\_СPU).

Программа не смогла запуститься с использованием CPU. Процессор не поддерживает OpenCL.

1. 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.

**Код программы:**

void Mul(const char\* filename, const cl\_context& context, cl\_device\_id\* devices, const cl\_command\_queue& commandQueue)

{

cl\_int status;

string sourceStr;

status = convertToString(filename, sourceStr);

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

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

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

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

const size\_t size = 100;

float input1[size];

float input2[size];

for (size\_t i = 1; i <= size; i++) {

input1[i-1] = (float)i;

input2[i-1] = (float)(i+1);

}

float\* output = (float\*)malloc(size);

cl\_mem inputBuffer = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, (size) \* sizeof(float), (void\*)input1, NULL);

cl\_mem inputBuffer2 = clCreateBuffer(context, CL\_MEM\_READ\_ONLY | CL\_MEM\_COPY\_HOST\_PTR, (size ) \* sizeof(float), (void\*)input2, NULL);

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

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

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

status = clSetKernelArg(kernel, 1, sizeof(cl\_mem), (void\*)&inputBuffer2);

status = clSetKernelArg(kernel, 2, sizeof(cl\_mem), (void\*)&outputBuffer);

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

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

status = clEnqueueReadBuffer(commandQueue, outputBuffer, CL\_TRUE, 0, size \* sizeof(float), output, 0, NULL, NULL);

for (size\_t i = 0; i < size; i++)

{

cout << output[i] << endl;

}

status = clReleaseKernel(kernel);

status = clReleaseProgram(program);

status = clReleaseMemObject(inputBuffer);

status = clReleaseMemObject(outputBuffer);

}

**Код программы OpenCL:**

\_\_kernel void mul( global float\* A,global float\* B, global float\* out)

{

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

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

}

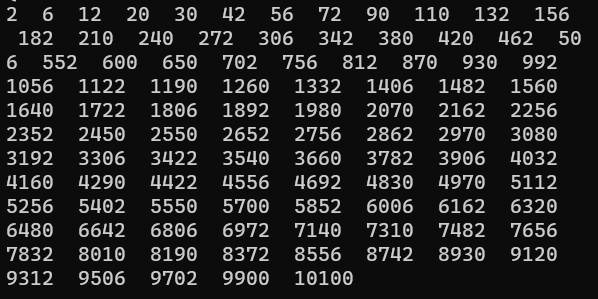


Рисунок 1 – Вывод программы умножения чисел

1. Получить сведения об устройстве (Имя устройства, тип, количество вычислительных блоков и т.п.).

**Код программы:**

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

printf(" Device number: %d\n", i);

char buffer[110];

status = clGetDeviceInfo(devices[i], CL\_DEVICE\_NAME, 100, buffer, NULL);

printf("Device name: %s\n", buffer);

// Запрос максимального количества вычислительных единиц устройства

cl\_uint UnitNum;

status = clGetDeviceInfo(devices[i], CL\_DEVICE\_MAX\_COMPUTE\_UNITS, sizeof(cl\_uint), &UnitNum, NULL);

printf("Compute Units Number: %d\n", UnitNum);

// Запрос частоты ядра устройства

cl\_uint frequency;

status = clGetDeviceInfo(devices[i], CL\_DEVICE\_MAX\_CLOCK\_FREQUENCY, sizeof(cl\_uint), &frequency, NULL);

printf("Device Frequency: %d(MHz)\n", frequency);

// Запрос общего объема памяти устройства

cl\_ulong GlobalSize;

status = clGetDeviceInfo(devices[i], CL\_DEVICE\_GLOBAL\_MEM\_SIZE, sizeof(cl\_ulong), &GlobalSize, NULL);

printf("Device Global Size: %0.0f(MB)\n", (float)GlobalSize / 1024 / 1024);

// Запрос строки глобального кэша памяти устройства

cl\_uint GlobalCacheLine;

status = clGetDeviceInfo(devices[i], CL\_DEVICE\_GLOBAL\_MEM\_CACHELINE\_SIZE, sizeof(cl\_uint), &GlobalCacheLine, NULL);

printf("Device Global CacheLine: %d(Byte)\n", GlobalCacheLine);

// Запрос версии OpenCL, поддерживаемой устройством

char DeviceVersion[110];

status = clGetDeviceInfo(devices[i], CL\_DEVICE\_VERSION, 100, DeviceVersion, NULL);

printf("Device Version:%s\n", DeviceVersion);

}

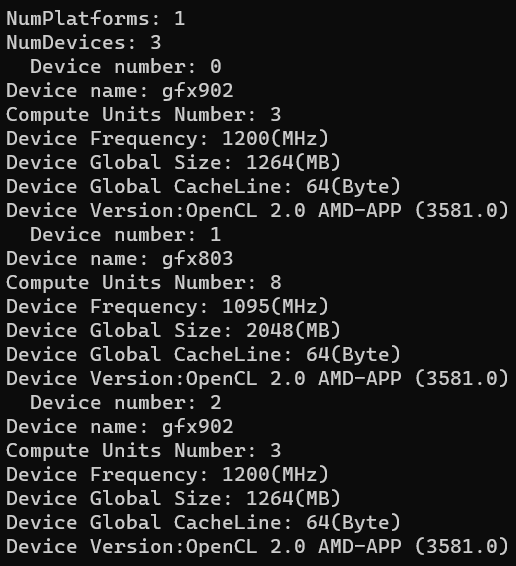
****

Рисунок 2 – Информация об устройствах

**Вывод**: в ходе работы были получены практические навыки программирования на OpenCL.