Романов А.Э.
А-14м-24
Лабораторная работа №6

Задание

Проверка простоты числа с использованием *CUDA*

Алгоритм

Входные данные

Число *a (uint64_t)*, над которым производится проверка.

Выходные данные

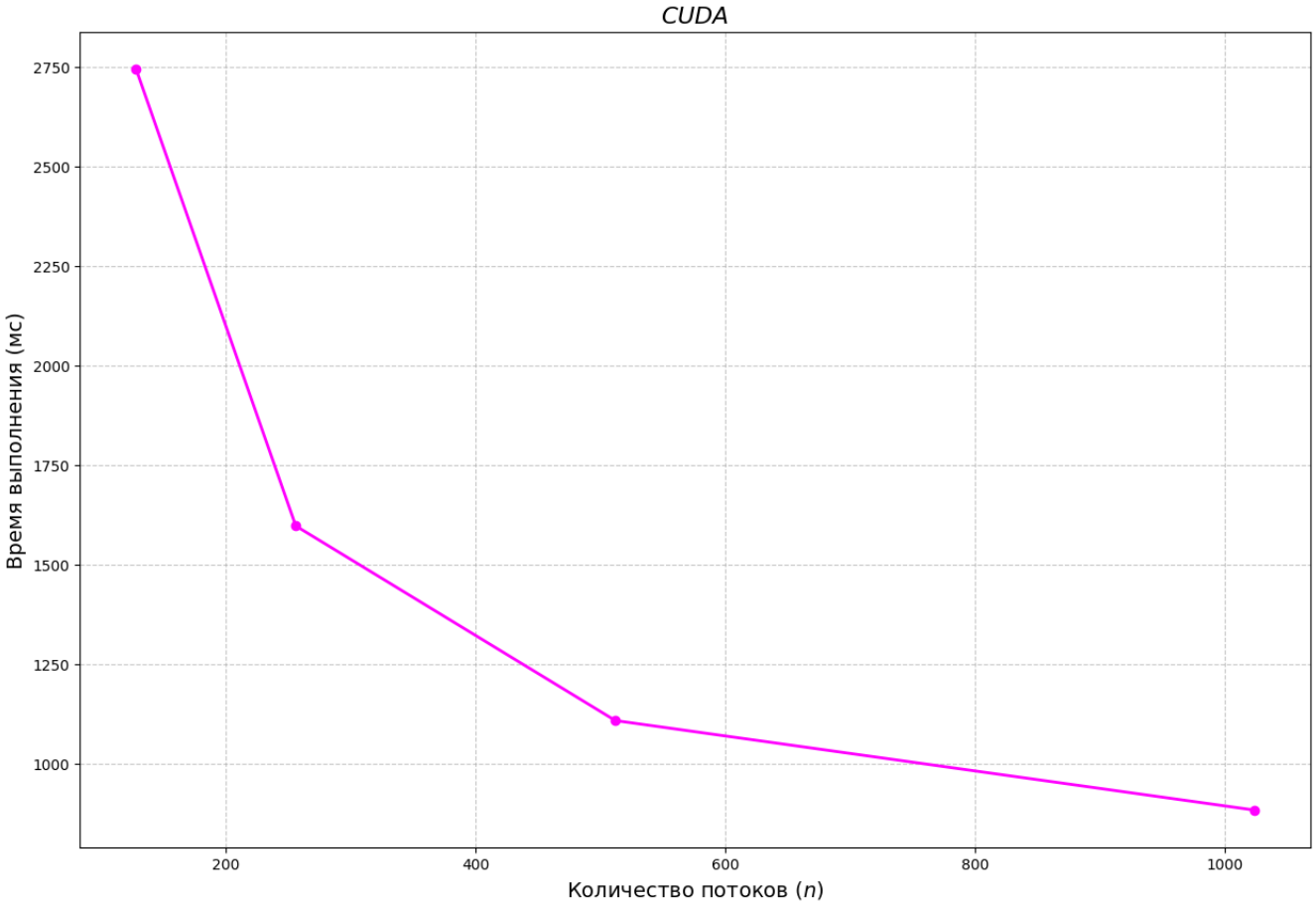
- Результат проверки;
- Потенциальный делитель
- Затраченное время (мс)

Алгоритм решения

Последовательный перебор делителей в отрезке $[2, \sqrt{a}]$ и проверка наличия остатка от деления. В случае, если остаток найден – алгоритм останавливается, дальнейшая проверка не требуется.

Тесты

N потоков	Число	Время, мс
1024	23058430092136939 51	883.715
512	23058430092136939 51	1108.43
256	23058430092136939 51	1597.08
128	23058430092136939 51	2744.67



Характеристики *GPU*

- *RTX* 3080;
- ядра *CUDA*: 8704;

- кол-во SM: 68;
- *Warp Size*: 32;

Программный код

```
#include <sstream>
#include <iostream>
#include <cuda_runtime.h>
#include <device_launch_parameters.h>

__device__ bool isprime(uint64_t n, uint64_t start, uint64_t end)
{
    for (uint64_t div = start; div <= end; div++) {
        if (n % div == 0) {
            return false;
        }
    }
    return true;
}

__global__ void isprime_kernel(uint64_t number, bool* result, uint64_t sqrtnum)
{
    uint64_t threadid = blockIdx.x * blockDim.x + threadIdx.x;
    uint64_t blocksize = (sqrtnum + blockDim.x - 1) / blockDim.x;
    uint64_t start = threadid * blocksize + 2;
    uint64_t end = start + blocksize - 1;
    if (start > sqrtnum) return;
    if (end > sqrtnum) end = sqrtnum;

    if (!isprime(number, start, end)) {
        *result = false;
    }
}

std::pair<bool, float> isprime_host(uint64_t number)
{
    if (number <= 1) return std::make_pair(false, 0);
    if (number <= 3) return std::make_pair(true, 0);
    if (number % 2 == 0 || number % 3 == 0) return  std::make_pair(false, 0);

    float elapsed = 0;
    cudaEvent_t timeBeg, timeEnd;
    cudaEventCreate(&timeBeg);
    cudaEventCreate(&timeEnd);

    uint64_t sqrtnum = sqrt(number);
    bool* d_result;
```

```
    cudaMalloc((void**)&d_result, sizeof(bool));
    cudaMemset(d_result, true, sizeof(bool));

    int threadsnum = 1024;
    int blocksnum = (sqrtnum + threadsnum - 1) / threadsnum;
    cudaEventRecord(timeBeg, 0);
    isprime_kernel << <blocksnum, threadsnum >> > (number, d_result, sqrtnum);
    cudaDeviceSynchronize();
    cudaEventRecord(timeEnd, 0);
    cudaEventSynchronize(timeEnd);
    cudaEventElapsedTime(&elapsed, timeBeg, timeEnd);

    bool result;
    cudaMemcpy(&result, d_result, sizeof(bool), cudaMemcpyDeviceToHost);
    cudaFree(d_result);
    cudaEventDestroy(timeBeg);
    cudaEventDestroy(timeEnd);

    return std::make_pair(result, elapsed);
}

int main(int argc, char* argv[])
{
    uint64_t number = 0;
    if (argc > 1) {
        std::stringstream sstream;
        sstream << argv[1];
        sstream >> number;
    }
    else {
        std::cout << "Enter number: ";
        std::cin >> number;
    }

    auto result = isprime_host(number);
    if (result.first) {
        std::cout << number << " is prime!\n";
    }
    else {
        std::cout << number << " is not prime!\n";
    }
    std::cout << "Time: " << result.second << " ms\n";
    system("pause");

    return 0;
}
```