

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КАФЕДРА ВС

Отчёт по лабораторной работе №3

«Оценка производительности

подсистемы памяти»

по дисциплине «Архитектура вычислительных систем»

Выполнил: студент гр. ИВ-823

Шиндель Э.Д.

Проверил: ст. преп. Кафедры ВС

Токмашева Е.И.

Новосибирск 2020

Содержание

Постановка задачи.....	3
Выполнение работы	7
Результат работы.....	10
Литература	13
Приложение	14

Постановка задачи

Задание: разработать программу (benchmark) для оценки производительности подсистемы памяти.

1. Написать программу (функцию) на языке C/C++/C# для оценки производительности подсистемы памяти.

На вход программы подать следующие аргументы.

- 1) Подсистема памяти. Предусмотреть возможность указать подсистему для проверки производительности: RAM (оперативная память), HDD/SSD и flash.
- 2) Размер блока данных в байтах, Кб или Мб. Если размерность не указана, то в байтах, если указана, то соответственно в Кбайтах или Мбайтах.
- 3) Число испытаний, т.е. число раз повторений измерений.

В качестве блока данных использовать одномерный массив, в котором произведение числа элементов на их размерность равна требуемому размеру блока данных. Массив инициализировать случайными значениями. Для тестирования HDD/SSD и flash создать в программе файлы в соответствующих директориях. Измерение времени реализовать с помощью функции `clock_gettime()` или аналогичной с точность до наносекунд. Измерять время исключительно на запись элемента в память или считывание из неё, без операций генерации или преобразования данных.

На выходе программы в одну строку CSV файла со следующей структурой: `[MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWriteTime;WriteBandwidth;AbsError(write);`

RelError(write);ReadTime;AverageReadTime;ReadBandwidthAbsError(read);RelError(read)], где

MemoryType – тип памяти (RAM|HDD|SSD|flash) или модель устройства, на котором проводятся испытания;

BlockSize – размер блока данных для записи и чтения на каждом испытании;

ElementType – тип элементов, используемых для заполнения массива данных;

BufferSize – размер буфера, т.е. порции данных для выполнения одной операции записи или чтения;

LaunchNum – порядковый номер испытания;

Timer – название функции обращения к таймеру (для измерения времени);

WriteTime – время выполнения отдельного испытания с номером LaunchNum [секунды];

AverageWriteTime – среднее время записи из LaunchNum испытаний [секунды];

WriteBandwidth – пропускная способность памяти ($BLOCK_SIZE / AverageWriteTime$) * 1E-6 [Mb/s];

AbsError(write) – абсолютная погрешность измерения времени записи или СКО [секунды];

RelError(write) – относительная погрешность измерения времени [%];

ReadTime – время выполнения отдельного испытания LaunchNum [секунды];

AverageReadTime – среднее время записи из LaunchNum испытаний [секунды];

ReadBandwidth – пропускная способность памяти ($\text{BLOCK_SIZE} / \text{AverageReadTime} \cdot 1\text{E-6}$ [Mb/s];

AbsError(read) – абсолютная погрешность измерения времени чтения или СКО [секунды];

RelError(read) – относительная погрешность измерения времени [%].

2. Написать программу(функцию) на языке C/C++/C# или скрипт (benchmark) реализующий серию испытаний программы(функции) из п.1. Оценить пропускную способность оперативной памяти при работе с блоками данных равными объёму кэш-линии, кэш-памяти L1, L2 и L3 уровня и превышающего его. Для HDD|SSD и flash провести серию из 20 испытаний с блоками данных начиная с 4 Мб с шагом 4Мб. Результаты всех испытаний сохранить в один CSV файл со структурой, описанной в п.1.

* Для HDD|SSD и flash оценить влияние размера буфера (BufferSize) на пропускную способность памяти.

3. На основе CSV файла построить сводные таблицы и диаграммы, отражающие:

1) Зависимость пропускной способности записи и чтения от размера блока данных (BlockSize) для разного типа памяти;

2) Зависимость погрешности измерения пропускной способности от размера блока данных для разного типа памяти;

3) Зависимость погрешности измерений от числа испытаний
LaunchNum;

4) * Зависимость пропускной способности памяти от размера буфера для
HDD|SSD и flash памяти;

Выполнение работы

Инструменты, используемые в ходе работы: bash, терминал, текстовый редактор, редактор исходного кода Visual Studio Code.

Основные шаги выполнения работы:

1. Реализация программы на языке Си.
2. Проведение тестов
3. Построение диаграмм.

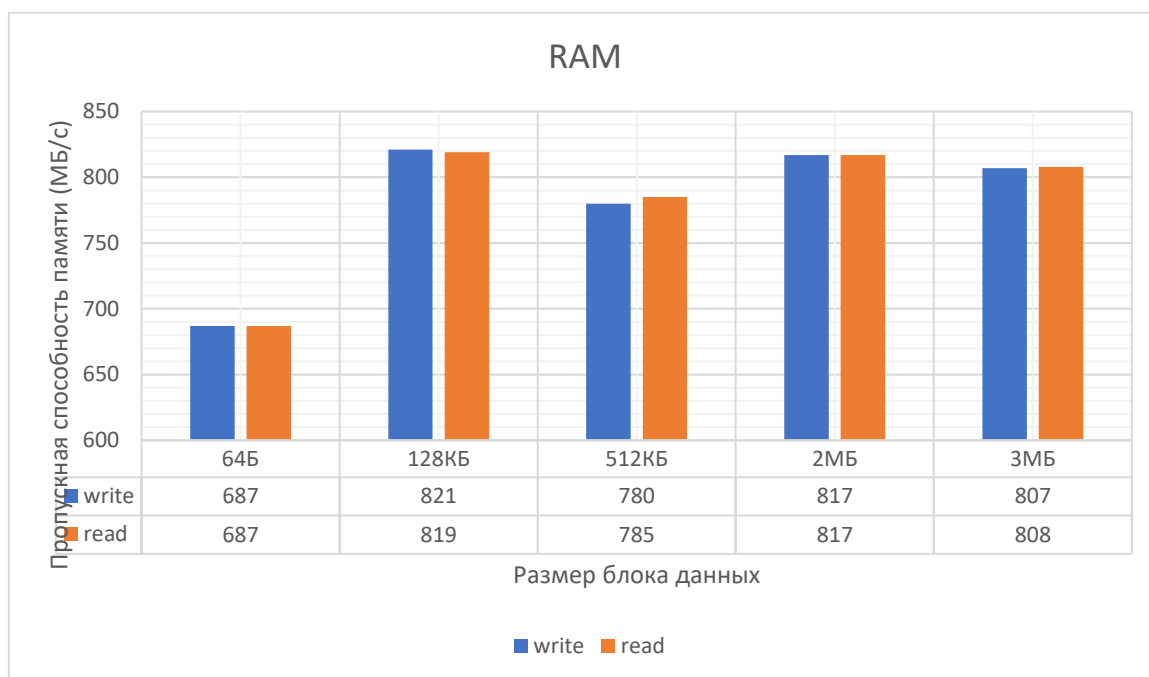


График 1. Зависимость пропускной способности записи и чтения от размера блока данных для типа памяти RAM.

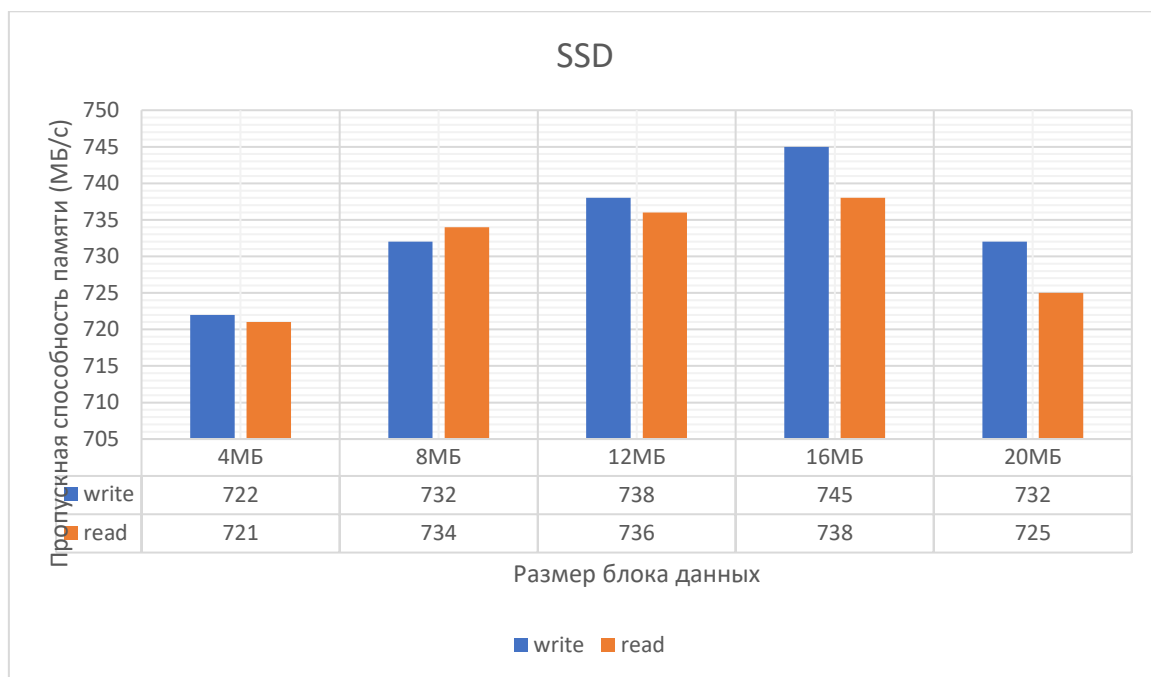


График 2. Зависимость пропускной способности записи и чтения от размера блока данных для типа памяти SSD.

Вывод: по итогам тестов мы видим, что для SSD при увеличении размера блока данных увеличивается и пропускная способность до какого-то предела. В данном примере до 16МБ, дальше идёт снижение.

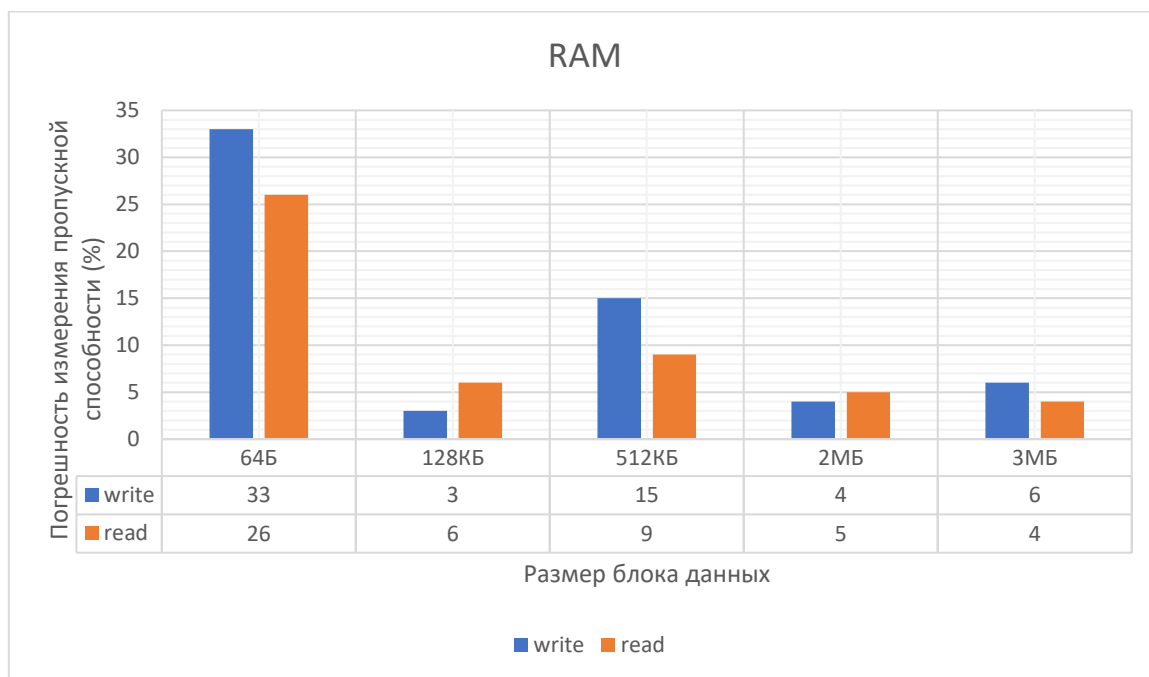


График 3. Зависимость погрешности измерения пропускной способности от размера блока данных для типа памяти RAM.

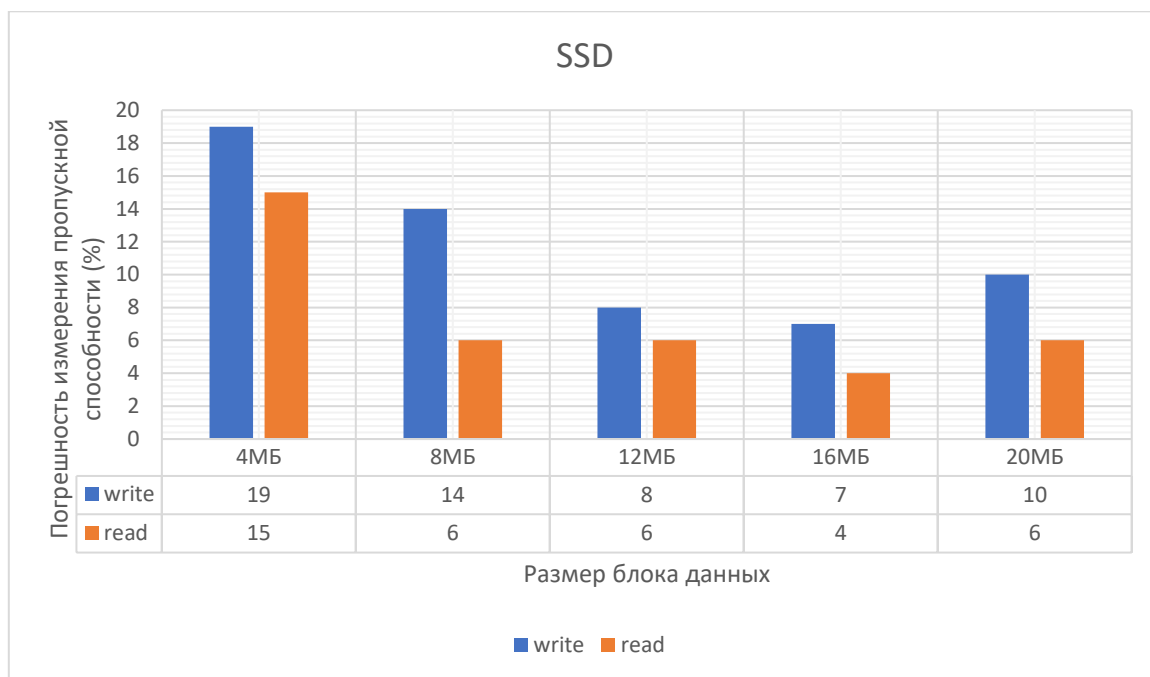


График 4. Зависимость погрешности измерения пропускной способности от размера блока данных для типа памяти SSD.

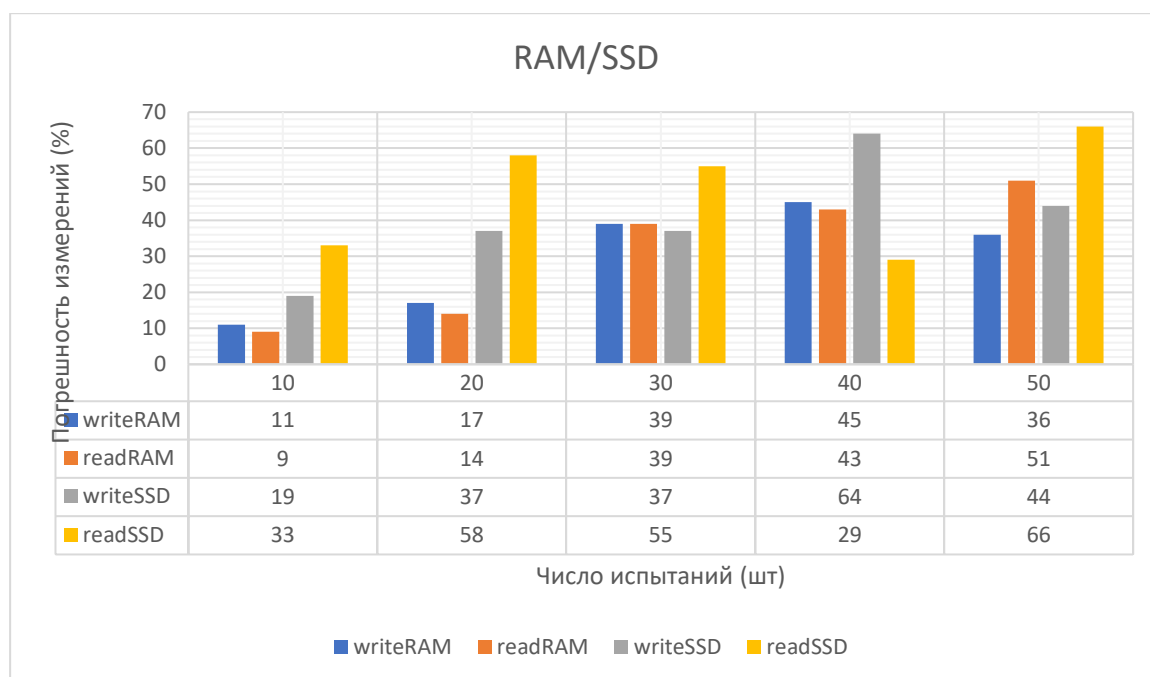
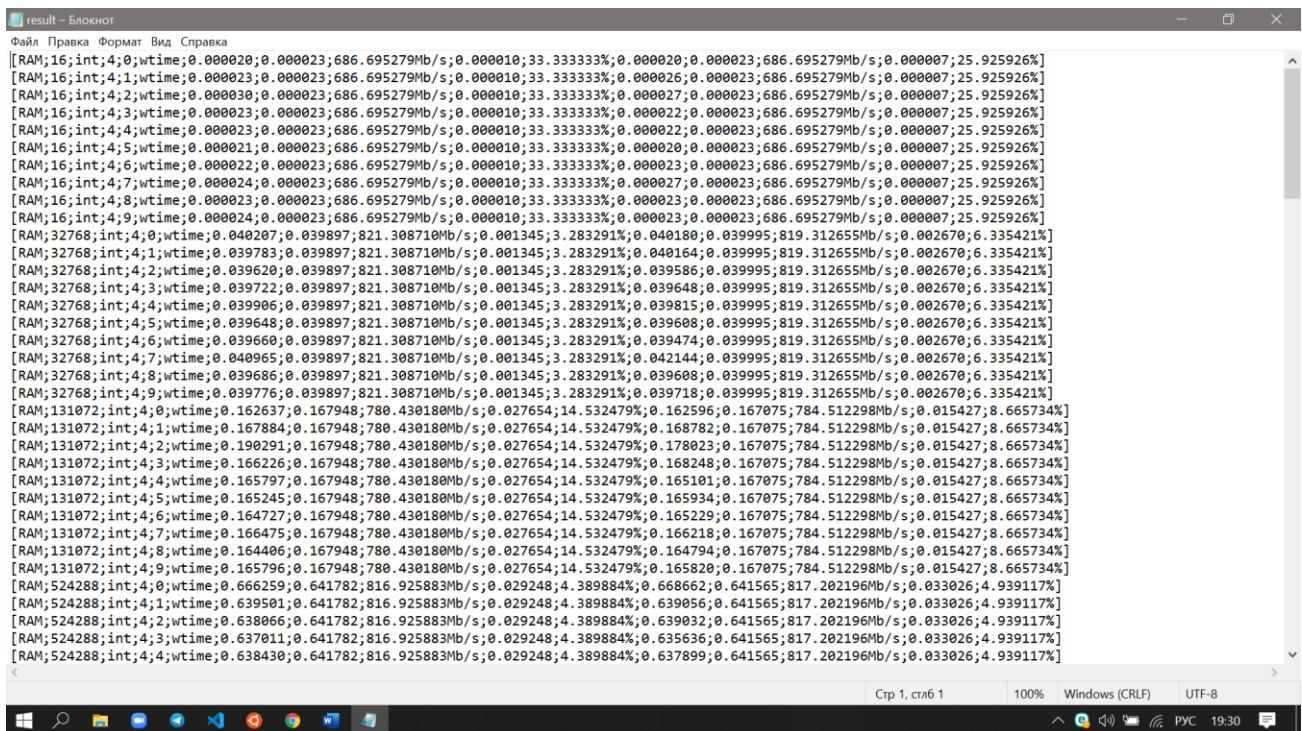


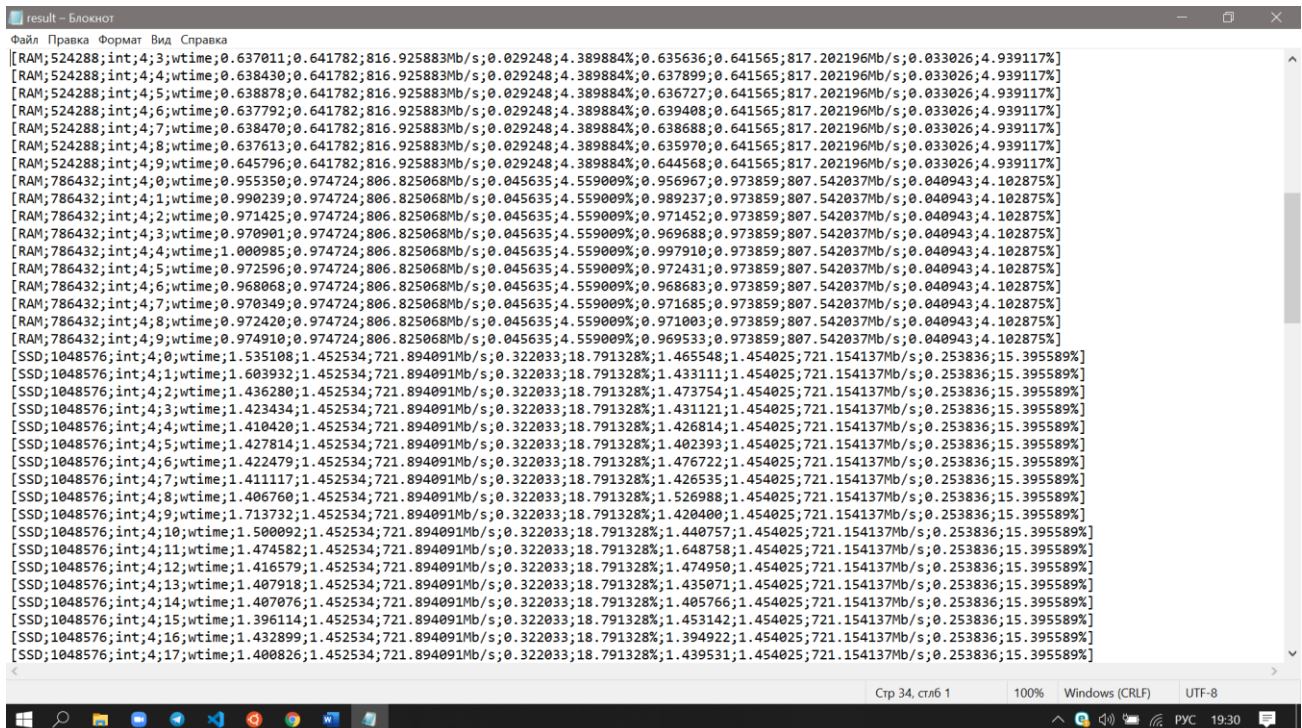
График 5. Зависимость погрешности измерений от числа испытаний.

Результат работы



```
File Edit Format View Help
[RAM;16;int;4;0;wtime;0.000020;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000020;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;1;wtime;0.000023;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000026;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;2;wtime;0.000030;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000027;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;3;wtime;0.000023;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000022;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;4;wtime;0.000023;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000022;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;5;wtime;0.000021;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000020;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;6;wtime;0.000022;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000023;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;7;wtime;0.000024;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000027;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;8;wtime;0.000023;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000023;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;16;int;4;9;wtime;0.000024;0.000023;686.695279Mb/s;0.000010;33.333333%;0.000023;0.000023;686.695279Mb/s;0.000007;25.925926%]
[RAM;32768;int;4;0;wtime;0.040207;0.039897;821.308710Mb/s;0.001345;3.283291%;0.040180;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;1;wtime;0.039783;0.039897;821.308710Mb/s;0.001345;3.283291%;0.040164;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;2;wtime;0.039620;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039586;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;3;wtime;0.039722;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039648;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;4;wtime;0.039906;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039815;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;5;wtime;0.039648;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039608;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;6;wtime;0.039660;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039474;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;7;wtime;0.040965;0.039897;821.308710Mb/s;0.001345;3.283291%;0.042144;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;8;wtime;0.039686;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039608;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;32768;int;4;9;wtime;0.039776;0.039897;821.308710Mb/s;0.001345;3.283291%;0.039718;0.039995;819.312655Mb/s;0.002670;6.335421%]
[RAM;131072;int;4;0;wtime;0.162637;0.167948;780.430180Mb/s;0.027654;14.532479%;0.162596;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;1;wtime;0.167884;0.167948;780.430180Mb/s;0.027654;14.532479%;0.168782;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;2;wtime;0.190291;0.167948;780.430180Mb/s;0.027654;14.532479%;0.178023;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;3;wtime;0.166226;0.167948;780.430180Mb/s;0.027654;14.532479%;0.168248;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;4;wtime;0.165797;0.167948;780.430180Mb/s;0.027654;14.532479%;0.165101;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;5;wtime;0.165245;0.167948;780.430180Mb/s;0.027654;14.532479%;0.165934;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;6;wtime;0.164727;0.167948;780.430180Mb/s;0.027654;14.532479%;0.165229;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;7;wtime;0.166475;0.167948;780.430180Mb/s;0.027654;14.532479%;0.166218;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;8;wtime;0.164406;0.167948;780.430180Mb/s;0.027654;14.532479%;0.164794;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;131072;int;4;9;wtime;0.165796;0.167948;780.430180Mb/s;0.027654;14.532479%;0.165820;0.167075;784.512298Mb/s;0.015427;8.665734%]
[RAM;524288;int;4;0;wtime;0.666259;0.641782;816.925883Mb/s;0.029248;4.389884%;0.668662;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;1;wtime;0.639501;0.641782;816.925883Mb/s;0.029248;4.389884%;0.639056;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;2;wtime;0.638066;0.641782;816.925883Mb/s;0.029248;4.389884%;0.639032;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;3;wtime;0.637011;0.641782;816.925883Mb/s;0.029248;4.389884%;0.635636;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;4;wtime;0.638430;0.641782;816.925883Mb/s;0.029248;4.389884%;0.637899;0.641565;817.202196Mb/s;0.033026;4.939117%]
```

Рисунок 1. Вывод результатов испытаний в csv файл.



```
File Edit Format View Help
[RAM;524288;int;4;3;wtime;0.637011;0.641782;816.925883Mb/s;0.029248;4.389884%;0.635636;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;4;wtime;0.638430;0.641782;816.925883Mb/s;0.029248;4.389884%;0.637899;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;5;wtime;0.638878;0.641782;816.925883Mb/s;0.029248;4.389884%;0.636727;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;6;wtime;0.637792;0.641782;816.925883Mb/s;0.029248;4.389884%;0.639408;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;7;wtime;0.638470;0.641782;816.925883Mb/s;0.029248;4.389884%;0.638688;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;8;wtime;0.637613;0.641782;816.925883Mb/s;0.029248;4.389884%;0.635970;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;524288;int;4;9;wtime;0.645796;0.641782;816.925883Mb/s;0.029248;4.389884%;0.644568;0.641565;817.202196Mb/s;0.033026;4.939117%]
[RAM;786432;int;4;0;wtime;0.955350;0.974724;806.825068Mb/s;0.045635;4.559009%;0.956967;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;1;wtime;0.990239;0.974724;806.825068Mb/s;0.045635;4.559009%;0.989237;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;2;wtime;0.971425;0.974724;806.825068Mb/s;0.045635;4.559009%;0.971452;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;3;wtime;0.970901;0.974724;806.825068Mb/s;0.045635;4.559009%;0.969688;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;4;wtime;1.000985;0.974724;806.825068Mb/s;0.045635;4.559009%;0.997910;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;5;wtime;0.972596;0.974724;806.825068Mb/s;0.045635;4.559009%;0.972431;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;6;wtime;0.968068;0.974724;806.825068Mb/s;0.045635;4.559009%;0.968683;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;7;wtime;0.970349;0.974724;806.825068Mb/s;0.045635;4.559009%;0.971685;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;8;wtime;0.972420;0.974724;806.825068Mb/s;0.045635;4.559009%;0.971003;0.973859;807.542037Mb/s;0.040943;4.102875%]
[RAM;786432;int;4;9;wtime;0.974910;0.974724;806.825068Mb/s;0.045635;4.559009%;0.969533;0.973859;807.542037Mb/s;0.040943;4.102875%]
[SSD;1048576;int;4;0;wtime;1.535108;1.452534;721.894091Mb/s;0.322033;18.791328%;1.465548;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;1;wtime;1.603932;1.452534;721.894091Mb/s;0.322033;18.791328%;1.433111;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;2;wtime;1.436280;1.452534;721.894091Mb/s;0.322033;18.791328%;1.473754;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;3;wtime;1.423434;1.452534;721.894091Mb/s;0.322033;18.791328%;1.431121;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;4;wtime;1.410420;1.452534;721.894091Mb/s;0.322033;18.791328%;1.426814;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;5;wtime;1.427814;1.452534;721.894091Mb/s;0.322033;18.791328%;1.402393;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;6;wtime;1.422479;1.452534;721.894091Mb/s;0.322033;18.791328%;1.476722;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;7;wtime;1.411117;1.452534;721.894091Mb/s;0.322033;18.791328%;1.426535;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;8;wtime;1.406760;1.452534;721.894091Mb/s;0.322033;18.791328%;1.526988;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;9;wtime;1.713732;1.452534;721.894091Mb/s;0.322033;18.791328%;1.420400;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;10;wtime;1.500092;1.452534;721.894091Mb/s;0.322033;18.791328%;1.440757;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;11;wtime;1.474582;1.452534;721.894091Mb/s;0.322033;18.791328%;1.648758;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;12;wtime;1.416579;1.452534;721.894091Mb/s;0.322033;18.791328%;1.474950;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;13;wtime;1.407918;1.452534;721.894091Mb/s;0.322033;18.791328%;1.435071;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;14;wtime;1.407076;1.452534;721.894091Mb/s;0.322033;18.791328%;1.405766;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;15;wtime;1.396114;1.452534;721.894091Mb/s;0.322033;18.791328%;1.453142;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;16;wtime;1.432899;1.452534;721.894091Mb/s;0.322033;18.791328%;1.394922;1.454025;721.154137Mb/s;0.253836;15.395589%]
[SSD;1048576;int;4;17;wtime;1.400826;1.452534;721.894091Mb/s;0.322033;18.791328%;1.439531;1.454025;721.154137Mb/s;0.253836;15.395589%]
```

Рисунок 2. Вывод результатов испытаний в csv файл

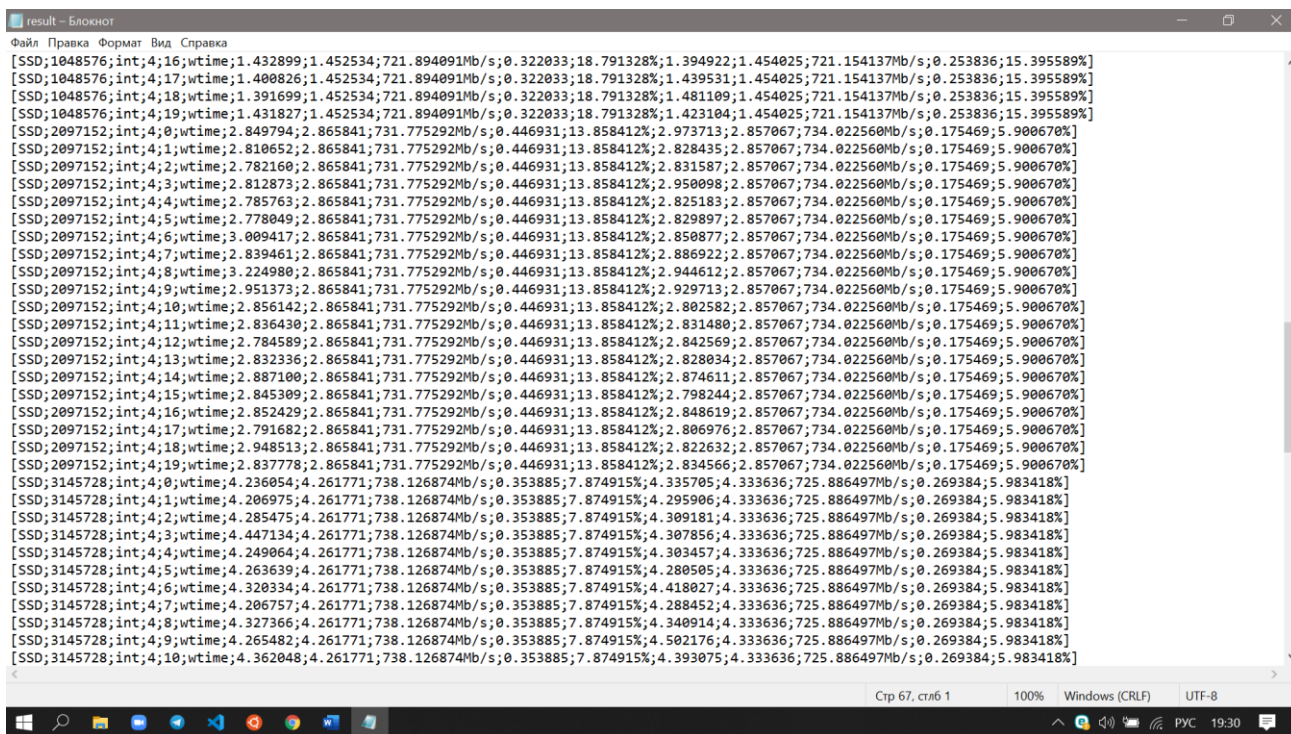


Рисунок 3. Вывод результатов испытаний в csv файл

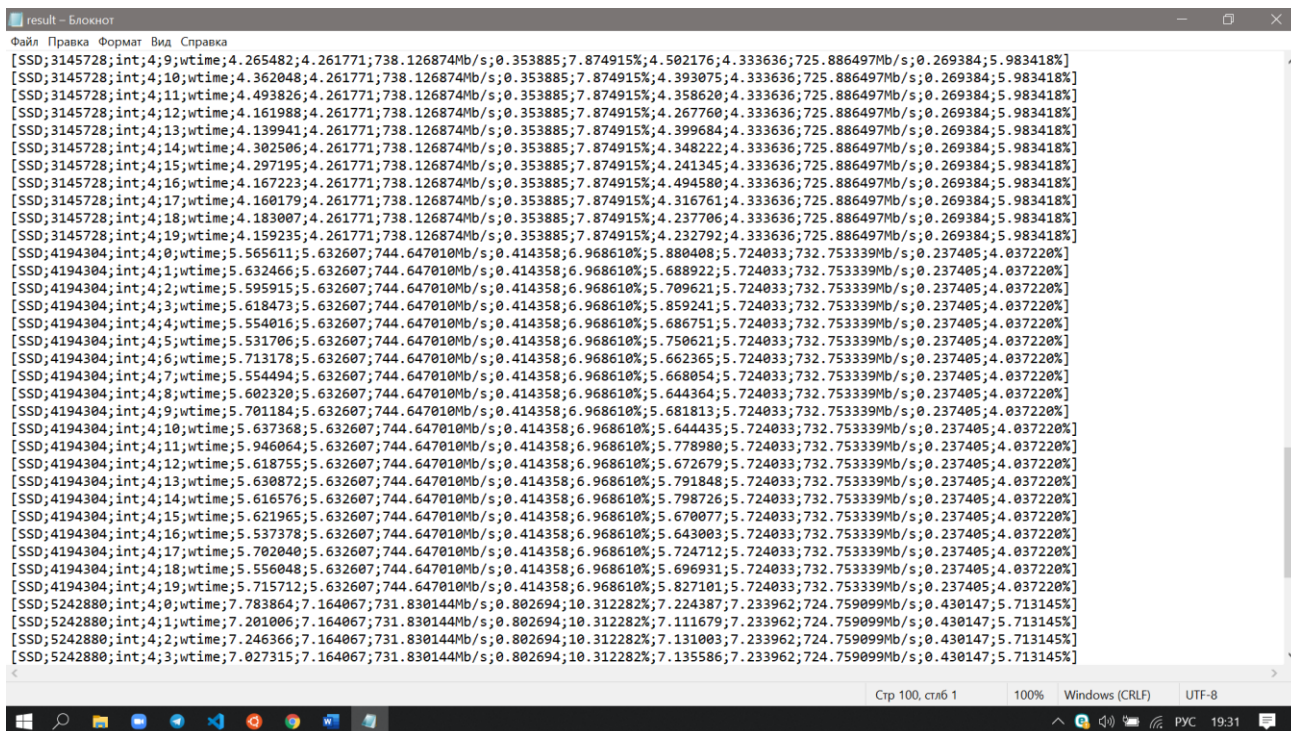


Рисунок 4. Вывод результатов испытаний в csv файл

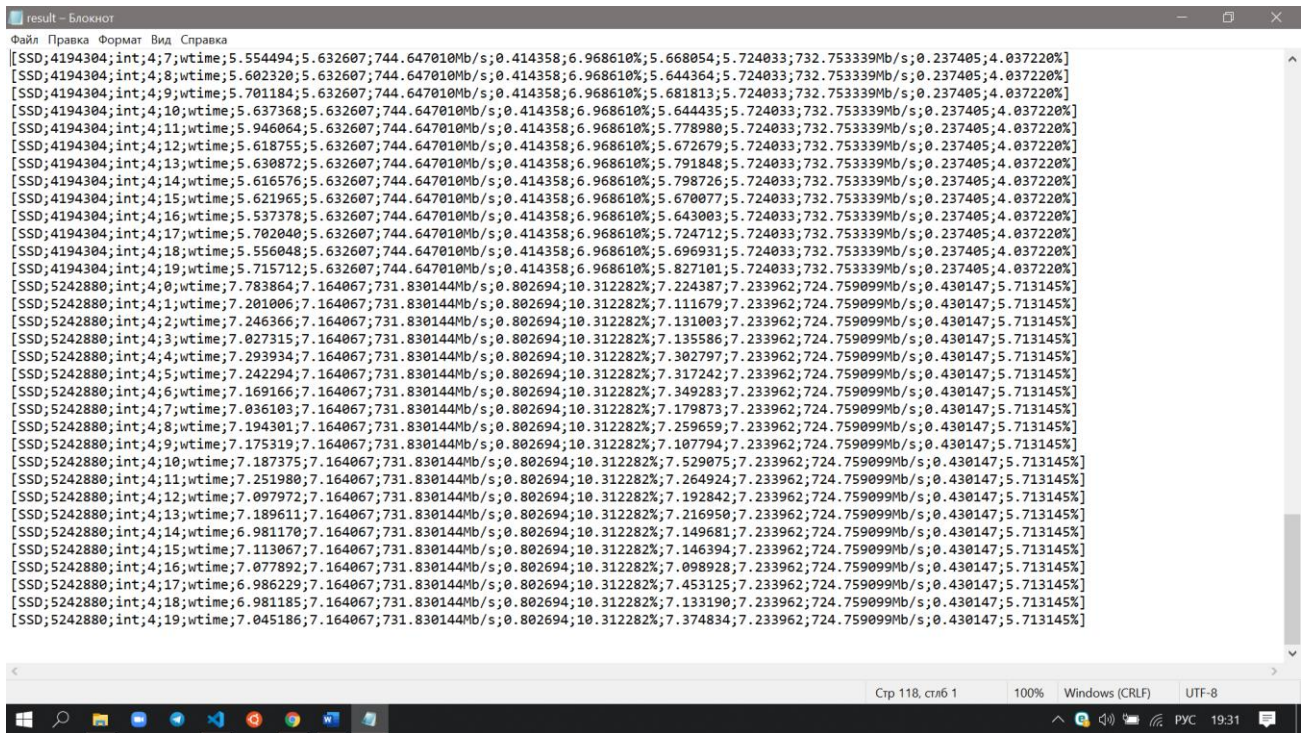


Рисунок 5. Вывод результатов испытаний в csv файл

Литература

1. [google.com](https://www.google.com)
2. [yandex.ru](https://www.yandex.ru)

Приложение

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/time.h>

double wtime()
{
    struct timeval t;
    gettimeofday(&t, NULL);
    return (double)t.tv_sec + (double)t.tv_usec * 1E-6;
}

void save_result(char *memory_type, int block_size, int launch_count)
{
    double middle_write = 0.0;
    double middle_read = 0.0;
    double AE_write, RE_write, AE_read, RE_read;
    double *time_write = malloc(sizeof(double) * launch_count);
    double *time_read = malloc(sizeof(double) * launch_count);
    int *num = malloc(sizeof(int) * launch_count);

    FILE *time = fopen("time.dat", "r");
    for (int i = 0; i < launch_count; i++) {
        fscanf(time, "%lf %lf %d", &time_write[i], &time_read[i], &num[i]);
        middle_write += time_write[i];
        middle_read += time_read[i];
    }
    fclose(time);

    middle_write /= launch_count;
    middle_read /= launch_count;

    double WB = (block_size / middle_write) * 1E-3;
    double RW = (block_size / middle_read) * 1E-3;

    double max_write = time_write[0];
    double min_write = time_write[0];
    double max_read = time_read[0];
    double min_read = time_read[0];
    for (int i = 1; i < launch_count; i++) {
        if (time_write[i] > max_write) max_write = time_write[i];
        if (time_write[i] < min_write) min_write = time_write[i];
        if (time_read[i] > max_read) max_read = time_read[i];
        if (time_read[i] < min_read) min_read = time_read[i];
    }
    AE_write = max_write - min_write;
    RE_write = AE_write / max_write * 100;
    AE_read = max_read - min_read;
    RE_read = AE_read / max_read * 100;

    FILE *result = fopen("result.csv", "a");
    for (int i = 0; i < launch_count; i++) {
        fprintf(result,
            "[%s;%d;int;4;%d;wtime;%lf;%lf;%lfMb/s;%lf;%lf%%;%lf;%lf;%lfMb/s;%lf;%lf%%]\n",
            memory_type, block_size, num[i],
            time_write[i], middle_write, WB, AE_write, RE_write,
            time_read[i], middle_read, RW, AE_read, RE_read);
    }
    fclose(result);
}
```

```

    free(time_write);
    free(time_read);
    free(num);
}

void save_time(double time_write, double time_read, int num)
{
    FILE *time = fopen("time.dat", "a");
    fprintf(time, "%f %f %d\n", time_write, time_read, num);
    fclose(time);
}

void test_ram(int block_size, int launch_count)
{
    int *memory = malloc(sizeof(int) * block_size);
    for (int i = 0; i < block_size; i++) {
        memory[i] = rand() % 201 - 100;
    }

    double time_read, time_write, sum_time_read, sum_time_write;
    for (int i = 0; i < launch_count; i++) {
        sum_time_read = 0.0;
        sum_time_write = 0.0;
        for (int j = 0; j < block_size; j++) {
            int random_number = rand() % 201 - 100;

            time_write = wtime();
            memory[j] = random_number;
            time_write = wtime() - time_write;

            time_read = wtime();
            random_number = memory[j];
            time_read = wtime() - time_read;

            sum_time_read += time_read;
            sum_time_write += time_write;
        }
        save_time(sum_time_write, sum_time_read, i);
    }
    free(memory);
}

void test_ssd(int block_size, int launch_count)
{
    double time_read, time_write, sum_time_read, sum_time_write;
    int *memory = malloc(sizeof(int) * block_size);

    for (int i = 0; i < block_size; i++) {
        memory[i] = rand() % 201 - 100;
    }

    for (int i = 0; i < launch_count; i++) {
        sum_time_write = 0.0;
        sum_time_read = 0.0;

        FILE *write = fopen("memory.dat", "w");
        for (int j = 0; j < block_size; j++) {
            time_write = wtime();
            fprintf(write, "%d ", memory[j]);
            time_write = wtime() - time_write;

            sum_time_write += time_write;

```

```

    }
    fprintf(write, "\n");
    fclose(write);

    FILE *read = fopen("memory.dat", "r");
    for (int j = 0; j < block_size; j++) {
        time_read = wtime();
        fscanf(read, "%d", &memory[j]);
        time_read = wtime() - time_read;

        sum_time_read += time_read;
    }
    fclose(read);
    save_time(sum_time_write, sum_time_read, i);
}
free(memory);
}

int main(int argc, char *argv[])
{
    if (argc != 7) {
        printf("ERROR! Not enough arguments\n");
        return 1;
    }

    int memory_type, block_size, launch_count;

    if ((!strcmp(argv[1], "-m")) || (!strcmp(argv[1], "--memory-type"))) {
        if (!strcmp(argv[2], "RAM")) memory_type = 0;
        else if (!strcmp(argv[2], "SSD")) memory_type = 1;
        else {
            printf("ERROR! Unknown argument: %s\n", argv[2]);
            return 1;
        }
    } else {
        printf("ERROR! Unknown key: %s\n", argv[1]);
        return 1;
    }

    if ((!strcmp(argv[3], "-b")) || (!strcmp(argv[3], "--block-size"))) {
        char arg0[10];
        strcpy(arg0, argv[4]);
        char *arg1 = strtok(argv[4], "/");
        if (strcmp(arg0, arg1)) {
            char *arg2 = strtok(NULL, "\\0");
            if (!strcmp(arg2, "1Kb")) block_size = atoi(arg1) * 1024 / sizeof(int);
            else if (!strcmp(arg2, "1Mb")) block_size = atoi(arg1) * 1024 * 1024 / sizeof(int);
            else {
                printf("ERROR! Unknown dimension: %s\n", arg2);
                return 1;
            }
        } else block_size = atoi(arg1) / sizeof(int);
        if (block_size <= 0) {
            printf("ERROR! Invalid block-size\n");
            return 1;
        }
    } else {
        printf("ERROR! Unknown key: %s\n", argv[3]);
        return 1;
    }

    if ((!strcmp(argv[5], "-l")) || (!strcmp(argv[5], "--launch-count"))) {

```



```

        launch_count = atoi(argv[6]);
        if (launch_count <= 0) {
            printf("ERROR! Wrong launch-count\n");
            return 1;
        }
    } else {
        printf("ERROR! Unknown key: %s\n", argv[5]);
        return 1;
    }

    srand(time(NULL));
    if (memory_type == 0) test_ram(block_size, launch_count);
    else test_ssd(block_size, launch_count);
    save_result(argv[2], block_size, launch_count);

    return 0;
}

```