

Федеральное государственное автономное  
образовательное учреждение  
высшего образования  
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

## ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа No 6. Программирование на языках ассемблера

тема

Преподаватель

\_\_\_\_\_

подпись, дата

А.С. Кузнецов  
инициалы, фамилия

Студент КИ18-17/16 031831229  
номер группы, зачетной книжки

\_\_\_\_\_

подпись, дата

В.А. Прекель  
инициалы, фамилия

Красноярск 2019

## СОДЕРЖАНИЕ

Содержание .....	2
1 Цель работы с постановкой задачи .....	4
1.1 Цель работы .....	4
1.2 Задача работы .....	4
1.3 Описание и пояснение к работе.....	4
2 Комментированный исходный код программ.....	6
2.1 Заголовочные файлы (описывают функции и из ассемблера, и из Си, благодаря ним можно использовать функции, написанные из ассемблера в тестах, написанных на Си) (корневой каталог).....	6
2.1.1. Matrix.h.....	6
2.1.2. MatrixIO.h.....	7
2.2 Реализация на ассемблере x86_32 (папка Lab_06_i386-gas) .....	7
2.2.1. Lab_06_i386-gas/main.s.....	7
2.2.2. Lab_06_i386-gas/Matrix.s .....	9
2.2.3. Lab_06_i386-gas/MatrixIO.s.....	13
2.3 Реализация на ассемблере MIPS32 (папка Lab_06_mips-spim) .....	17
2.3.1. Lab_06_mips-spim/Lab_06_mips-spim.s .....	17
2.4 Реализация на ассемблере AArch64 (папка Lab_06_aarch64-gas).....	25
2.4.1. Lab_06_aarch64-gas/main.s .....	25
2.4.2. Lab_06_aarch64-gas/Matrix.s .....	26
2.4.3. Lab_06_aarch64-gas/MatrixIO.s .....	28
2.5 Реализация на Си (папка Lab_06_C) .....	31
2.5.1. Lab_06_C/main.c.....	31

2.5.2. Lab_06_C/Matrix.c .....	31
2.5.3. Lab_06_C/MatrixIO.c .....	33
3 Тестовые примеры работы программ .....	33
3.1 Пример сборки и работы под x86_32 .....	33
3.1.1. Сборка и запуск основной программы на ассемблере (WSL Ubuntu 18.04) .....	33
3.1.2. Сборка и запуск основной программы на ассемблере (Ubuntu 16.04) .....	34
3.1.3. Сборка и запуск модульных тестов (WSL Ubuntu 18.04) .....	35
3.2 Пример сборки и работы под AArch64 .....	36
3.2.1. Сборка и запуск основной программы на ассемблере (WSL Ubuntu 18.04) .....	36
3.2.2. Очистка тестов от тестов для x86_32, сборка и запуск тестов (WSL Ubuntu 18.04) .....	37
3.2.3. Сборка, запуск основной программы на ассемблере и тестов (Android) .....	38
3.3 Запуск основной программы под MIPS32 в симуляторе SPIM .....	39
3.4 Пример сборки и работы реализации на Си .....	40
3.4.1. Сборка и запуск (WSL Ubuntu 18.04) .....	40
3.5 Сборка и запуск тестов (WSL Ubuntu 18.04) .....	40

## **1 Цель работы с постановкой задачи**

### **1.1 Цель работы**

Разработка программ на языках ассемблера.

### **1.2 Задача работы**

Требуется разработать ассемблерную программу, исходный код которой представляет собой программу, разделенную на основную часть и подпрограммы (не менее двух). Результат вычислений выводится на экран. Целевые вычислительные системы x86\_32 и MIPS32, а также (по желанию студента) другие (например, ARM).

В последнем случае должны быть описаны средства проверки корректности — онлайн и/или симуляторы, наподобие SPIM, или компиляторы наподобие gas. Функционально корректная дополнительная реализация ассемблерного кода вознаграждается одним бонусным баллом, добавляемым к оценке за обязательную реализацию. Еще один бонусный балл добавляется студенту, представившему методический материал с описанием целевой вычислительной архитектуры и особенностях программирования на языке ассемблера.

**Вариант 4.** Дана целочисленная матрица размера  $M \times N$ . Найти количество ее строк и столбцов, все элементы которых различны.

### **1.3 Описание и пояснение к работе**

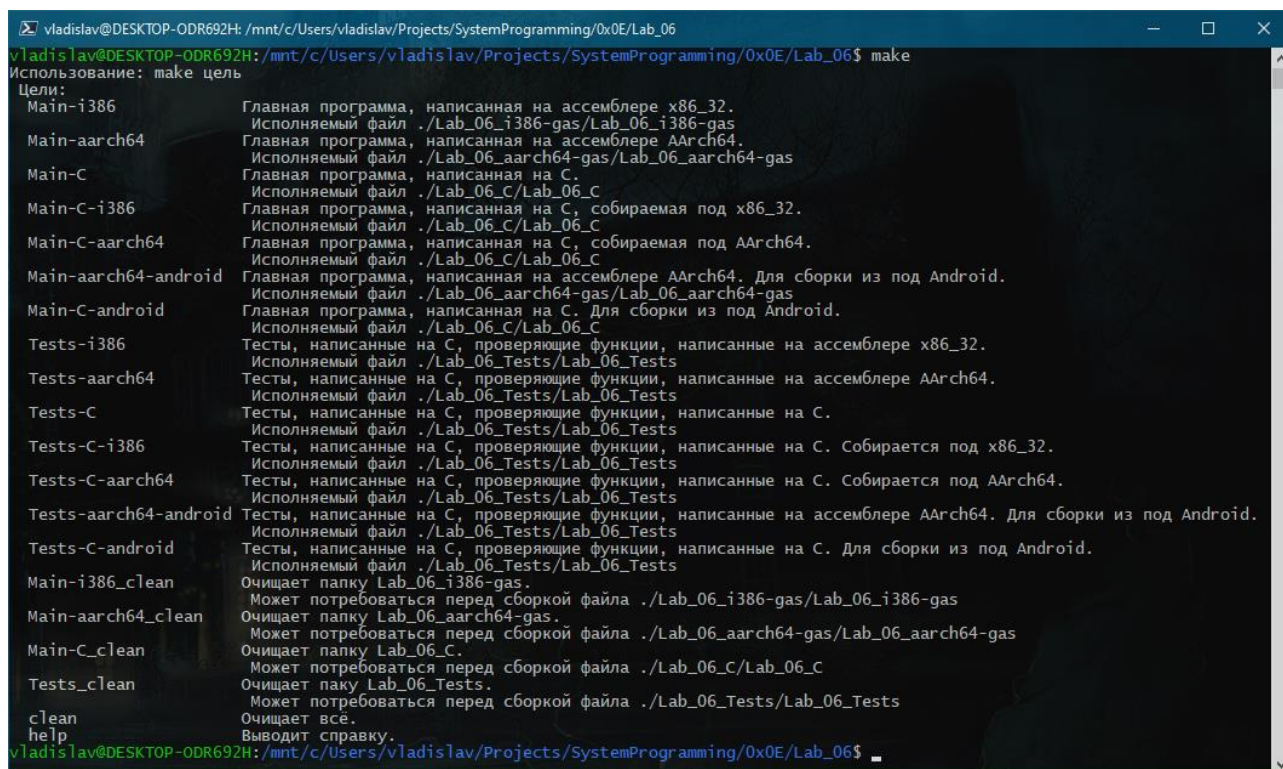
Работа выполнена для трёх ассемблеров: x86\_32, MIPS32 и дополнительно AArch64. Так же написана реализация на Си и модульные тесты на Си, которые могут проверять как и реализацию на Си, так и на ассемблере x86\_32 или

AArch64. Используется система сборки GNU Make. Процесс сборки показан в тестовых примерах работы программ. Дополнительно для ассемблера AArch64 и реализации на Си предусмотрена сборка на операционной системе Android с помощью эмулятора терминала Termux.

Для сборки и запуска могут понадобиться пакеты:

- gcc-multilib
- spim
- gcc-aarch64-linux-gnu
- qemu-user

Сборка ведётся через корневой Makefile. Для получения списка целей и пояснения к ним требуется ввести просто make.



```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make
Использование: make цель
Цели:
Main-i386          Главная программа, написанная на ассемблере x86_32.
                   Исполняемый файл ./Lab_06_i386-gas/Lab_06_i386-gas
Main-aarch64       Главная программа, написанная на ассемблере AArch64.
                   Исполняемый файл ./Lab_06_aarch64-gas/Lab_06_aarch64-gas
Main-C             Главная программа, написанная на С.
                   Исполняемый файл ./Lab_06_C/Lab_06_C
Main-C-i386        Главная программа, написанная на С, собираемая под x86_32.
                   Исполняемый файл ./Lab_06_C/Lab_06_C
Main-C-aarch64     Главная программа, написанная на С, собираемая под AArch64.
                   Исполняемый файл ./Lab_06_C/Lab_06_C
Main-aarch64-android Главная программа, написанная на ассемблере AArch64. Для сборки из под Android.
                   Исполняемый файл ./Lab_06_aarch64-gas/Lab_06_aarch64-gas
Main-C-android     Главная программа, написанная на С. Для сборки из под Android.
                   Исполняемый файл ./Lab_06_C/Lab_06_C
Tests-i386         Тесты, написанные на С, проверяющие функции, написанные на ассемблере x86_32.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Tests-aarch64      Тесты, написанные на С, проверяющие функции, написанные на ассемблере AArch64.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Tests-C            Тесты, написанные на С, проверяющие функции, написанные на С.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Tests-C-i386       Тесты, написанные на С, проверяющие функции, написанные на С. Собирается под x86_32.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Tests-C-aarch64    Тесты, написанные на С, проверяющие функции, написанные на С. Собирается под AArch64.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Tests-aarch64-android Тесты, написанные на С, проверяющие функции, написанные на ассемблере AArch64. Для сборки из под Android.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Tests-C-android    Тесты, написанные на С, проверяющие функции, написанные на С. Для сборки из под Android.
                   Исполняемый файл ./Lab_06_Tests/Lab_06_Tests
Main-i386_clean    Очищает папку Lab_06_i386-gas.
                   Может потребоваться перед сборкой файла ./Lab_06_i386-gas/Lab_06_i386-gas
Main-aarch64_clean Очищает папку Lab_06_aarch64-gas.
                   Может потребоваться перед сборкой файла ./Lab_06_aarch64-gas/Lab_06_aarch64-gas
Main-C_clean       Очищает папку Lab_06_C.
                   Может потребоваться перед сборкой файла ./Lab_06_C/Lab_06_C
Tests_clean        Очищает паку Lab_06_Tests.
                   Может потребоваться перед сборкой файла ./Lab_06_Tests/Lab_06_Tests
clean              Очищает все.
help              Выводит справку.
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 1 – Вывод справки

## 2 Комментированный исходный код программ.

**2.1 Заголовочные файлы (описывают функции и из ассемблера, и из Си, благодаря ним можно использовать функции, написанные из ассемблера в тестах, написанных на Си) (корневой каталог)**

### 2.1.1. Matrix.h

```
/// \file
/// \brief Функции, подсчитывающие матрицу
/// \details Функции, подсчитывающие матрицу.

#ifndef MATRIX_H
#define MATRIX_H

#include <stdbool.h>

/// Проверяет ряд или строку матрицы, хранящейся в одномерном массиве
/// на различие всех элементов.
///
/// \param pArray Указатель на первый элемент ряда или строки
/// \param step 1 для строки, кол-во столбцов (n) для столбца.
/// \param size Кол-во столбцов (n) для строки, кол-во строк (m) для столбца.
/// \return Истина если все элементы различны.
bool CheckAllDifferent(int* pArray, int step, int size);

/// Подсчитывает кол-во строк в матрице, хранящейся в одномерном массиве,
/// в которых все элементы различны.
///
/// \param pMatrix Указатель на первый элемент массива, в котором хранится
/// матрица.
/// \param m Кол-во строк.
/// \param n Кол-во столбцов.
/// \return Кол-во строк в матрице, все элементы которых различны.
int CountDifferentLines(int* pMatrix, int m, int n);

/// Подсчитывает кол-во столбцов в матрице, хранящейся в одномерном массиве,
/// в которых все элементы различны.
///
/// \param pMatrix Указатель на первый элемент массива, в котором хранится
/// матрица.
/// \param m Кол-во строк.
/// \param n Кол-во столбцов.
/// \return Кол-во столбцов в матрице, все элементы которых различны.
int CountDifferentRows(int* pMatrix, int m, int n);

#endif //MATRIX_H
```

## 2.1.2. MatrixIO.h

```
/// \file
/// \brief Функции для ввода и вывода матрицы, хранящейся в одномерном массиве
/// \details Функции для ввода и вывода матрицы, хранящейся в одномерном массиве.

#ifndef MATRIXIO_H
#define MATRIXIO_H

/// Считывает кол-во строк и столбцов, выводя сообщение о том, что
/// считывается.
///
/// \param pM Указатель на кол-во строк.
/// \param pN Указатель на кол-во столбцов.
void ReadMN(int* pM, int* pN);

/// Считывает матрицу, перед каждым элементом выводя сообщение о том,
/// какой элемент считывается.
///
/// \param pMatrix Указатель на матрицу.
/// \param m Кол-во строк.
/// \param n Кол-во столбцов.
void ReadMatrix(int* pMatrix, int m, int n);

/// Выводит матрицу, перед этим выводит сколько в матрице строк и столбцов.
///
/// \param pMatrix Указатель на матрицу.
/// \param m Кол-во строк.
/// \param n Кол-во столбцов.
void WriteMatrix(int* pMatrix, int m, int n);

#endif //MATRIXIO_H
```

## 2.2 Реализация на ассемблере x86\_32 (папка Lab\_06\_i386-gas)

### 2.2.1. Lab\_06\_i386-gas/main.s

```
# Вариант 4.
# Дана целочисленная матрица размера M x N.
# Найти количество ее строк и столбцов, все элементы которых различны.
.text

# int main()
.globl main
main:
    pushl %ebp
    movl %esp, %ebp
    pushl %esi
    subl $28, %esp

    # Занятый стек 40 байт:
    # 4(%ebp) [4] old %ebp
```

```

# 0(%ebp) [4] old %esi
# -4(%ebp) [4]
# -8(%ebp) [4]
# -12(%ebp) [4] m
# -16(%ebp) [4] n
# -20(%ebp) [4] pMatrix
# -24(%ebp) [4] 8(%esp)
# -28(%ebp) [4] 4(%esp)
# -32(%ebp) [4] 0(%esp)

# 4(%esp) [4] &n
# 0(%esp) [4] &m
leal -12(%ebp), %eax
movl %eax, (%esp)
leal -16(%ebp), %eax
movl %eax, 4(%esp)
calll ReadMN

# 0(%esp) [4] m * n * 4
movl -12(%ebp), %eax
imull -16(%ebp), %eax
imull $4, %eax
movl %eax, (%esp)
calll malloc
movl %eax, -20(%ebp)

# 8(%esp) [4] n
# 4(%esp) [4] m
# 0(%esp) [4] pMatrix
movl -12(%ebp), %ecx
movl -16(%ebp), %edx
movl %eax, (%esp)
movl %ecx, 4(%esp)
movl %edx, 8(%esp)
calll ReadMatrix

# 0(%esp) [4] &NewLine1
leal NewLine1, %eax
movl %eax, (%esp)
calll printf

# 8(%esp) [4] n
# 4(%esp) [4] m
# 0(%esp) [4] pMatrix
movl -20(%ebp), %ecx
movl -12(%ebp), %edx
movl -16(%ebp), %esi
movl %ecx, (%esp)
movl %edx, 4(%esp)
movl %esi, 8(%esp)
calll WriteMatrix

# 8(%esp) [4] n
# 4(%esp) [4] m
# 0(%esp) [4] pMatrix
movl -20(%ebp), %eax
movl -12(%ebp), %ecx
movl -16(%ebp), %edx
movl %eax, (%esp)
movl %ecx, 4(%esp)

```



```

movl %edx, 8(%esp)
calll CountDifferentLines

# 4(%esp) [4] CountDifferentLines()
# 0(%esp) [4] &MessageCountLinesFormat1
leal MessageCountLinesFormat1, %ecx
movl %ecx, (%esp)
movl %eax, 4(%esp)
calll printf

# 8(%esp) [4] n
# 4(%esp) [4] m
# 0(%esp) [4] pMatrix
movl -20(%ebp), %ecx
movl -12(%ebp), %edx
movl -16(%ebp), %esi
movl %ecx, (%esp)
movl %edx, 4(%esp)
movl %esi, 8(%esp)
calll CountDifferentRows

# 4(%esp) [4] CountDifferentRows()
# 0(%esp) [4] &MessageCountLinesFormat1
leal MessageCountRowsFormat1, %ecx
movl %ecx, (%esp)
movl %eax, 4(%esp)
calll printf

# 0(%esp) [4] pMatrix
movl -20(%ebp), %ecx
movl %ecx, (%esp)
calll free

# return 0
movl $0, %eax

addl $28, %esp
popl %esi
popl %ebp
retl

```

```

NewLine1:
.asciz "\n"

```

```

MessageCountLinesFormat1:
.asciz "Кол-во строк, все элементы которых различны: %d\n"

```

```

MessageCountRowsFormat1:
.asciz "Кол-во столбцов, все элементы которых различны: %d\n"

```

## 2.2.2. Lab\_06\_i386-gas/Matrix.s

```
.text
```

```
# bool CheckAllDifferent(int* pArray, int step, int size)
```

```

.globl CheckAllDifferent
CheckAllDifferent:
    pushl %ebp
    movl %esp, %ebp
    subl $20, %esp

    # Занятый стек 40 байт:
    # 16(%ebp) [4] size
    # 12(%ebp) [4] step
    # 8(%ebp) [4] pArray
    # 4(%ebp) [4] old %ebp
    # 0(%ebp) [4]
    # -4(%ebp) [4]
    # -5(%ebp) [1] result
    # -6(%ebp) [1]
    # -7(%ebp) [1]
    # -8(%ebp) [1]
    # -12(%ebp) [4] max = step * size
    # -16(%ebp) [4] i
    # -20(%ebp) [4] j

    # -12(%ebp) <- 16(%ebp) * 12(%ebp)
    movl 12(%ebp), %eax
    mull 16(%ebp)
    movl %eax, -12(%ebp)

    # for (-16(%ebp) <- 0; -16(%ebp) < -12(%ebp); -16(%ebp) += 12(%ebp))
    Loop1_Start:
        # -16(%ebp) <- 0
        movl $0, -16(%ebp)
        jmp Loop1_Check
    Loop1_Body:
        # for (-20(%ebp) <- -16(%ebp) + 12(%ebp); -20(%ebp) < -12(%ebp); -20(%ebp)
        += 12(%ebp))
        Loop2_Start:
            # -20(%ebp) <- -16(%ebp) + 12(%ebp)
            movl -16(%ebp), %eax
            addl 12(%ebp), %eax
            movl %eax, -20(%ebp)
            jmp Loop2_Check
        Loop2_Body:
            # %eax <- 8(%ebp)[-16(%ebp)]
            movl 8(%ebp), %eax
            movl -16(%ebp), %ecx
            movl (%eax,%ecx,4), %eax
            # ;if %eax != 8(%ebp)[-20(%ebp)] goto Loop2_Continue
            movl 8(%ebp), %ecx
            movl -20(%ebp), %edx
            cmpl (%ecx,%edx,4), %eax
            jne Loop2_Continue
            # -5(%ebp) <- $0
            movb $0, -5(%ebp)
            jmp Return1
        Loop2_Continue:
            # -20(%ebp) += 12(%ebp)
            movl 12(%ebp), %eax
            addl %eax, -20(%ebp)
        Loop2_Check:
            # ;if -20(%ebp) < -12(%ebp) goto Loop2_Body
            movl -20(%ebp), %eax

```

```

        cmpl -12(%ebp), %eax
        j1 Loop2_Body
        # -16(%ebp) += 12(%ebp)
        movl 12(%ebp), %eax
        addl %eax, -16(%ebp)
Loop1_Check:
        # ;if -16(%ebp) < -12(%ebp) goto Loop1_Body
        movl -16(%ebp), %eax
        cmpl -12(%ebp), %eax
        j1 Loop1_Body

# -5(%ebp) <- $1
movb $1, -5(%ebp)

Return1:
# %al <- -5(%ebp)
# return -5(%ebp)
movb -5(%ebp), %al

addl $20, %esp
popl %ebp
retl

# int CountDifferentLines(int* pMatrix, int m, int n)
.globl CountDifferentLines
CountDifferentLines:
pushl %ebp
movl %esp, %ebp
subl $24, %esp

# Занятый стек 44 байт:
# 16(%ebp) [4] n
# 12(%ebp) [4] m
# 8(%ebp) [4] pMatrix
# 4(%ebp) [4] old %ebp
# 0(%ebp) [4]
# -4(%ebp) [4] c
# -8(%ebp) [4] i
# -9(%ebp) [1] check
# -10(%ebp) [1]
# -11(%ebp) [1]
# -12(%ebp) [1]
# -16(%ebp) [4] 8(%esp) n
# -20(%ebp) [4] 4(%esp) 1
# -24(%ebp) [4] 0(%esp) pMatrix + i * n

# -4(%ebp) <- 0
movl $0, -4(%ebp)

# for (-8(%ebp) <- 0; -8(%ebp) < 12(%ebp); -8(%ebp)++)
Loop3_Start:
        # -8(%ebp) <- 0
        movl $0, -8(%ebp)
        jmp Loop3_Check
Loop3_Body:
        movl 8(%ebp), %eax
        movl -8(%ebp), %ecx
        imull 16(%ebp), %ecx
        imull $4, %ecx

```

```

    addl %ecx, %eax
    movl 16(%ebp), %ecx
    movl %eax, (%esp)
    movl $1, 4(%esp)
    movl %ecx, 8(%esp)
    calll CheckAllDifferent
    andb $1, %al
    movb %al, -9(%ebp)
    testb $1, -9(%ebp)
    je Loop3_Continue
    movl -4(%ebp), %eax
    addl $1, %eax
    movl %eax, -4(%ebp)
Loop3_Continue:
    movl -8(%ebp), %eax
    addl $1, %eax
    movl %eax, -8(%ebp)
Loop3_Check:
    movl -8(%ebp), %eax
    cmpl 12(%ebp), %eax
    jl Loop3_Body

# %eax <- -4(%ebp)
movl -4(%ebp), %eax

addl $24, %esp
popl %ebp
retl

```

```

# int CountDifferentRows(int* pMatrix, int m, int n)

```

```

.globl CountDifferentRows

```

```

CountDifferentRows:

```

```

    pushl %ebp
    movl %esp, %ebp
    subl $24, %esp

```

```

# Занятый стек 44 байт:
# 16(%ebp) [4] n
# 12(%ebp) [4] m
# 8(%ebp) [4] pMatrix
# 4(%ebp) [4] old %ebp
# 0(%ebp) [4]
# -4(%ebp) [4] c
# -8(%ebp) [4] i
# -9(%ebp) [1] check
# -10(%ebp) [1]
# -11(%ebp) [1]
# -12(%ebp) [1]
# -16(%ebp) [4] 8(%esp) m
# -20(%ebp) [4] 4(%esp) n
# -24(%ebp) [4] 0(%esp) pMatrix + i

```

```

# -4(%ebp) <- 0
movl $0, -4(%ebp)

```

```

Loop4_Start:
    movl $0, -8(%ebp)
    jmp Loop4_Check
Loop4_Body:

```

```

    movl 8(%ebp), %eax
    movl -8(%ebp), %ecx
    imull $4, %ecx
    addl %ecx, %eax
    movl 16(%ebp), %ecx
    movl 12(%ebp), %edx
    movl %eax, (%esp)
    movl %ecx, 4(%esp)
    movl %edx, 8(%esp)
    calll CheckAllDifferent
    andb $1, %al
    movb %al, -9(%ebp)
    testb $1, -9(%ebp)
    je Loop4_Continue
    movl -4(%ebp), %eax
    addl $1, %eax
    movl %eax, -4(%ebp)
Loop4_Continue:
    movl -8(%ebp), %eax
    addl $1, %eax
    movl %eax, -8(%ebp)
Loop4_Check:
    movl -8(%ebp), %eax
    cmpl 16(%ebp), %eax
    jl Loop4_Body

# %eax < -4(%ebp)
movl -4(%ebp), %eax

addl $24, %esp
popl %ebp
retl

```

### 2.2.3. Lab\_06\_i386-gas/MatrixIO.s

```

.text

# void ReadMN(int* pM, int* pN)
.globl ReadMN
ReadMN:
    pushl %ebp
    movl %esp, %ebp

    # Занятый стек 16 байт:
    # 12(%ebp) [4] pN
    # 8(%ebp) [4] pM
    # 4(%ebp) [4] old %ebp
    # 0(%ebp) [4]

    # Занятый стек 16 байт перед вызовом printf:
    # 12(%ebp) [4] pN
    # 8(%ebp) [4] pM
    # 4(%ebp) [4] old %ebp
    # 0(%ebp) [4] 0(%esp) &InputMNMessage1

    leal InputMNMessage1, %edx

```

```

pushl %edx
calll printf
addl $4, %esp

# Занятый стек 24 байт перед вызовом scanf:
# 12(%ebp) [4] pN
# 8(%ebp) [4] pM
# 4(%ebp) [4] old %ebp
# 0(%ebp) [4] 8(%esp) pN
# -4(%ebp) [4] 4(%esp) pM
# -8(%ebp) [4] 0(%esp) &InputMNFormat1

pushl 12(%ebp)
pushl 8(%ebp)
leal InputMNFormat1, %edx
pushl %edx
calll scanf
addl $12, %esp

popl %ebp
retl

# void ReadMatrix(int* pMatrix, int m, int n)
.globl ReadMatrix
ReadMatrix:
pushl %ebp
movl %esp, %ebp
subl $24, %esp

# Занятый стек 40 байт:
# 16(%ebp) [4] n
# 12(%ebp) [4] m
# 8(%ebp) [4] pMatrix
# 4(%ebp) [4] old %ebp
# 0(%ebp) [4]
# -4(%ebp) [4] i
# -8(%ebp) [4] j
# -12(%ebp) [4]
# -16(%ebp) [4]
# -20(%ebp) [4]
# -24(%ebp) [4] 0(%esp)

Loop3_Start:
movl $0, -4(%ebp)
jmp Loop3_Check
Loop3_Body:
Loop4_Start:
movl $0, -8(%ebp)
jmp Loop4_Check
Loop4_Body:
movl -4(%ebp), %eax
movl -8(%ebp), %ecx
leal InputMessageFormat1, %edx
movl %edx, (%esp)
movl %eax, 4(%esp)
movl %ecx, 8(%esp)
calll printf

movl 8(%ebp), %ecx

```

```

        movl -4(%ebp), %edx
        imull 16(%ebp), %edx
        addl -8(%ebp), %edx
        imull $4, %edx
        addl %edx, %ecx
        leal InputFormat1, %edx
        movl %edx, (%esp)
        movl %ecx, 4(%esp)
        calll scanf

        movl -8(%ebp), %eax
        addl $1, %eax
        movl %eax, -8(%ebp)
Loop4_Check:
        movl -8(%ebp), %eax
        cmpl 16(%ebp), %eax
        jnl Loop4_Body
        movl -4(%ebp), %eax
        addl $1, %eax
        movl %eax, -4(%ebp)
Loop3_Check:
        movl -4(%ebp), %eax
        cmpl 12(%ebp), %eax
        jnl Loop3_Body

        addl $24, %esp
        popl %ebp
        retl

```

```

# void WriteMatrix(int* pMatrix, int m, int n)

```

```

        .globl WriteMatrix

```

```

WriteMatrix:

```

```

        pushl %ebp
        movl %esp, %ebp
        pushl %ebx
        pushl %edi
        pushl %esi
        subl $20, %esp

```

```

# Занятый стек 48 байт:
# 16(%ebp) [4] n
# 12(%ebp) [4] m
# 8(%ebp) [4] pMatrix
# 4(%ebp) [4] old %ebp
# 0(%ebp) [4] old %ebx
# -4(%ebp) [4] old %edi
# -8(%ebp) [4] old %esi
# -12(%ebp) [4]
# -16(%ebp) [4] i
# -20(%ebp) [4] j
# -24(%ebp) [4] 4(%esp)
# -28(%ebp) [4] 0(%esp)

```

```

        leal OutputFormatMatrix1, %ebx
        push 16(%ebp)
        push 12(%ebp)
        push %ebx
        calll printf
        addl $16, %esp

```

```

# for (-16(%ebp) <- 0; -16(%ebp) < 12(%ebp); -16(%ebp)++)
Loop1_Start:
    movl $0, -16(%ebp)
    jmp Loop1_Check
Loop1_Body:
    # for (-20(%ebp) <- 0; -20(%ebp) < 16(%ebp); -20(%ebp)++)
    Loop2_Start:
        movl $0, -20(%ebp)
        jmp Loop2_Check
    Loop2_Body:
        movl 8(%ebp), %eax
        movl -16(%ebp), %ecx
        imull 16(%ebp), %ecx
        addl -20(%ebp), %ecx
        movl (%eax,%ecx,4), %eax
        leal OutputFormat1, %ecx
        movl %ecx, (%esp)
        movl %eax, 4(%esp)
        calll printf
        movl -20(%ebp), %eax
        addl $1, %eax
        movl %eax, -20(%ebp)
    Loop2_Check:
        movl -20(%ebp), %eax
        cmpl 16(%ebp), %eax
        jl Loop2_Body
    leal NewLine, %eax
    movl %eax, (%esp)
    calll printf
    movl -16(%ebp), %eax
    addl $1, %eax
    movl %eax, -16(%ebp)
Loop1_Check:
    movl -16(%ebp), %eax
    cmpl 12(%ebp), %eax
    jl Loop1_Body

    addl $16, %esp
    popl %esi
    popl %edi
    popl %ebx
    popl %ebp
    retl

```

InputMNMessage1:

```
.asciz "Введите М и N (кол-во строк и столбцов, a[M][N]): "
```

InputMNFormat1:

```
.asciz "%d%d"
```

InputMessageFormat1:

```
.asciz "Введите a[%d][%d]: "
```

InputFormat1:

```
.asciz "%d"
```

OutputFormatMatrix1:

```
.asciz "Матрица a[%d][%d]:\n"
```



```

OutputFormat1:
    .asciz "%d "

NewLine:
    .asciz "\n"

```

## 2.3 Реализация на ассемблере MIPS32 (папка Lab\_06\_mips-spm)

### 2.3.1. Lab\_06\_mips-spm/Lab\_06\_mips-spm.s

```

# Вариант 4.
# Дана целочисленная матрица размера M x N.
# Найдти количество ее строк и столбцов, все элементы которых различны.
    .text

# bool CheckAllDifferent(int* pArray, int step, int size)
    .globl CheckAllDifferent
CheckAllDifferent:
    mul $t3, $a1, $a2                # $t3 <- $a1 * $a2

    Loop5_Start:
        li $t4, 0                    # $t4 <- 0
        j Loop5_Check                # goto Loop5_Check
    Loop5_Body:
        Loop6_Start:
            add $t5, $t4, $a1          # $t5 <- $t4 + $a1
            j Loop6_Check              # goto Loop6_Check
        Loop6_Body:
            li $t6, 4                  # $t6 <- 4
            mul $t6, $t6, $t4          # $t6 *= $t4
            add $t6, $t6, $a0          # $t6 += $a0
            lw $t6, ($t6)              # $t6 <- *t6
            li $t7, 4                  # $t7 <- 4
            mul $t7, $t7, $t5          # $t7 *= $t5
            add $t7, $t7, $a0          # $t7 += $a0
            lw $t7, ($t7)              # $t7 <- *t7
            bne $t6, $t7, Loop6_Cont   # if $t6 != $t7 goto Loop6_Cont
            li $v0, 0                  # return false
            jr $ra                      # go back to $ra
        Loop6_Cont:
            add $t5, $t5, $a1          # $t5 += $a1
    Loop6_Check:
        blt $t5, $t3, Loop6_Body      # if $t5 < $t3 goto Loop6_Body
        add $t4, $t4, $a1             # $t4 += $a1
    Loop5_Check:
        blt $t4, $t3, Loop5_Body      # if $t4 < $t3 goto Loop5_Body

    li $v0, 1                          # return true
    jr $ra                             # go back to $ra

# int CountDifferentLines(int* pMatrix, int m, int n)
    .globl CountDifferentLines

```

CountDifferentLines:

```

addi $sp, $sp, -4
sw $ra, ($sp)
addi $sp, $sp, -4
sw $s0, ($sp)
addi $sp, $sp, -4
sw $s1, ($sp)
addi $sp, $sp, -4
sw $s2, ($sp)
addi $sp, $sp, -4
sw $s3, ($sp)
addi $sp, $sp, -4
sw $s4, ($sp)
addi $sp, $sp, -4
sw $s5, ($sp)

```

```

move $s0, $a0
move $s1, $a1
move $s2, $a2

```

```

# $s0 <- $a0
# $s1 <- $a1
# $s2 <- w2

```

```
li $s3, 0
```

```
# $s3 <- 0
```

```
li $s5, 0
```

```
# $s5 <- 0
```

*# for (\$s4 = 0, \$s4 < \$s1, \$s4++)*

Loop7\_Start:

```
li $s4, 0
```

```
# $s4 <- 0
```

```
j Loop7_Check
```

```
# goto Loop7_Check
```

Loop7\_Body:

```
li $a0, 4
```

```
# $a0 <- 4
```

```
mul $a0, $a0, $s3
```

```
# $a0 <- $s3 * 4
```

```
add $a0, $a0, $s0
```

```
# $a0 += $s0
```

```
li $a1, 1
```

```
# $a1 <- 1
```

```
move $a2, $s2
```

```
# $a2 <- $s2
```

```
jal CheckAllDifferent
```

```
# call CheckAllDifferent
```

```
beq $v0, $zero, Loop7_Continue
```

```
# if $a0 == false goto Loop7_Continue
```

```
add $s5, $s5, 1
```

```
# $s5++
```

Loop7\_Continue:

```
add $s3, $s3, $s2
```

```
# $s3 += $s2
```

```
add $s4, $s4, 1
```

```
# $s4++
```

Loop7\_Check:

```
blt $s4, $s1, Loop7_Body
```

```
# if $s4 < $s1 goto Loop7_Body
```

```
move $v0, $s5
```

```
# return $s5
```

```
lw $s5, ($sp)
```

```
addi $sp, $sp, 4
```

```
lw $s4, ($sp)
```

```
addi $sp, $sp, 4
```

```
lw $s3, ($sp)
```

```
addi $sp, $sp, 4
```

```
lw $s2, ($sp)
```

```
addi $sp, $sp, 4
```

```
lw $s1, ($sp)
```

```
addi $sp, $sp, 4
```

```
lw $s0, ($sp)
```

```
addi $sp, $sp, 4
```

```
lw $ra, ($sp)
```

```
addi $sp, $sp, 4
```

```
jr $ra
```

```
# go back to $ra
```

```

# int CountDifferentRows(int* pMatrix, int m, int n)
.globl CountDifferentRows
CountDifferentRows:
    addi $sp, $sp, -4
    sw $ra, ($sp)
    addi $sp, $sp, -4
    sw $s0, ($sp)
    addi $sp, $sp, -4
    sw $s1, ($sp)
    addi $sp, $sp, -4
    sw $s2, ($sp)
    addi $sp, $sp, -4
    sw $s3, ($sp)
    addi $sp, $sp, -4
    sw $s4, ($sp)
    addi $sp, $sp, -4
    sw $s5, ($sp)

    move $s0, $a0                # $s0 <- $a0
    move $s1, $a1                # $s1 <- $a1
    move $s2, $a2                # $s2 <- w2

    li $s5, 0                    # $s5 <- 0

    # for ($s4 = 0, $s4 < $s2, $s4++)
    Loop8_Start:
        li $s4, 0                # $s4 <- 0
        j Loop8_Check            # goto Loop8_Check
    Loop8_Body:
        li $a0, 4                # $a0 <- 4
        mul $a0, $a0, $s4        # $a0 <- $s4 * 4
        add $a0, $a0, $s0        # $a0 += $s0
        move $a1, $s2            # $a1 <- $s2
        move $a2, $s1            # $a2 <- $s1
        jal CheckAllDifferent    # call CheckAllDifferent
        beq $v0, $zero, Loop8_Continue # if $a0 == false goto Loop8_Continue
        add $s5, $s5, 1          # $s5++
    Loop8_Continue:
        add $s4, $s4, 1          # $s4++
    Loop8_Check:
        blt $s4, $s2, Loop8_Body # if $s4 < $s2 goto Loop8_Body

    move $v0, $s5                # $a0 <- $s5

    lw $s5, ($sp)
    addi $sp, $sp, 4
    lw $s4, ($sp)
    addi $sp, $sp, 4
    lw $s3, ($sp)
    addi $sp, $sp, 4
    lw $s2, ($sp)
    addi $sp, $sp, 4
    lw $s1, ($sp)
    addi $sp, $sp, 4
    lw $s0, ($sp)
    addi $sp, $sp, 4
    lw $ra, ($sp)
    addi $sp, $sp, 4

```

```

    jr $ra                                # go back to $ra

# void ReadMN(int* pM, int* pN)
.globl ReadMN
ReadMN:
    addi $sp, $sp, -4
    sw $ra, ($sp)
    addi $sp, $sp, -4
    sw $s0, ($sp)
    addi $sp, $sp, -4
    sw $s1, ($sp)
    addi $sp, $sp, -4
    sw $s2, ($sp)
    addi $sp, $sp, -4
    sw $s3, ($sp)
    addi $sp, $sp, -4
    sw $s4, ($sp)

    move $s0, $a0                          # $s0 <- $a0
    move $s1, $a1                          # $s1 <- $a1

    li $v0, 4                             # $v0 <- 4
    la $a0, InputMNMessage1               # $a0 <- &InputMNMessage1
    syscall                               # syscall print_string

    li $v0, 5                             # $v0 <- 5
    syscall                               # syscall read_int

    sw $v0, ($s0)                         # *$s0 <- $v0

    li $v0, 5                             # $v0 <- 5
    syscall                               # syscall read_int

    sw $v0, ($s1)                         # *$s1 <- $v0

    lw $s4, ($sp)
    addi $sp, $sp, 4
    lw $s3, ($sp)
    addi $sp, $sp, 4
    lw $s2, ($sp)
    addi $sp, $sp, 4
    lw $s1, ($sp)
    addi $sp, $sp, 4
    lw $s0, ($sp)
    addi $sp, $sp, 4
    lw $ra, ($sp)
    addi $sp, $sp, 4

    jr $ra                                # go back to $ra

# void ReadMatrix(int* pMatrix, int m, int n)
.globl ReadMatrix
ReadMatrix:
    addi $sp, $sp, -4
    sw $ra, ($sp)
    addi $sp, $sp, -4
    sw $s0, ($sp)

```

```

addi $sp, $sp, -4
sw $s1, ($sp)
addi $sp, $sp, -4
sw $s2, ($sp)
addi $sp, $sp, -4
sw $s3, ($sp)
addi $sp, $sp, -4
sw $s4, ($sp)

move $s0, $a0          # $s0 <- $a0
move $s1, $a1          # $s1 <- $a1
move $s2, $a2          # $s2 <- $a2

# for ($s3 <- 0; $s3 < $s1; $s3++)
Loop3_Start:
    li $s3, 0          # $s3 <- 0
    j Loop3_Check      # goto Loop3_Check
Loop3_Body:
    # for ($s4 <- 0; $s4 < $s2; $s4++)
    Loop4_Start:
        li $s4, 0      # $s4 <- 0
        j Loop4_Check  # goto Loop4_Check
    Loop4_Body:
        li $v0, 4
        la $a0, InputMessagePart1
        syscall
        li $v0, 1
        move $a0, $s3
        syscall
        li $v0, 4
        la $a0, InputMessagePart2
        syscall
        li $v0, 1
        move $a0, $s4
        syscall
        li $v0, 4
        la $a0, InputMessagePart3
        syscall
        li $v0, 5          # $v0 <- 5
        syscall          # syscall read_int
        move $t3, $v0
        mul $t0, $s3, $s2  # $t0 <- $s3 * $s2
        add $t0, $t0, $s4  # $t0 += $s4
        li $t1, 4          # $t1 <- 4
        mul $t0, $t0, $t1  # $t0 *= 4
        add $t0, $t0, $s0  # $t0 += $s0
        sw $t3, ($t0)      # *t0 <- $a0

        addi $s4, $s4, 1    # $s4++
    Loop4_Check:
        blt $s4, $s2, Loop4_Body # if $s4 < $s2 goto Loop4_Body
        addi $s3, $s3, 1      # $s3++
Loop3_Check:
    blt $s3, $s1, Loop3_Body    # if $s3 < $s1 goto Loop3_Body

lw $s4, ($sp)
addi $sp, $sp, 4
lw $s3, ($sp)
addi $sp, $sp, 4

```

```

lw $s2, ($sp)
addi $sp, $sp, 4
lw $s1, ($sp)
addi $sp, $sp, 4
lw $s0, ($sp)
addi $sp, $sp, 4
lw $ra, ($sp)
addi $sp, $sp, 4

jr $ra

```

```
# void WriteMatrix(int* pMatrix, int m, int n)
```

```
.globl WriteMatrix
```

```
WriteMatrix:
```

```

addi $sp, $sp, -4
sw $ra, ($sp)
addi $sp, $sp, -4
sw $s0, ($sp)
addi $sp, $sp, -4
sw $s1, ($sp)
addi $sp, $sp, -4
sw $s2, ($sp)
addi $sp, $sp, -4
sw $s3, ($sp)
addi $sp, $sp, -4
sw $s4, ($sp)

```

```

move $s0, $a0           # $s0 <- $a0
move $s1, $a1           # $s1 <- $a1
move $s2, $a2           # $s2 <- $a2

```

```

li $v0, 4
la $a0, OutputFormatMatrixPart1
syscall
li $v0, 1
move $a0, $s1
syscall
li $v0, 4
la $a0, OutputFormatMatrixPart2
syscall
li $v0, 1
move $a0, $s2
syscall
li $v0, 4
la $a0, OutputFormatMatrixPart3
syscall

```

```
# for ($s3 <- 0; $s3 < $s1; $s3++)
```

```
Loop1_Start:
```

```

    li $s3, 0           # $s3 <- 0
    j  Loop1_Check      # goto Loop1_Check

```

```
Loop1_Body:
```

```
# for ($s4 <- 0; $s4 < $s2; $s4++)
```

```
Loop2_Start:
```

```

    li $s4, 0           # $s4 <- 0
    j  Loop2_Check      # goto Loop2_Check

```

```
Loop2_Body:
```

```

    li $v0, 1           # $v0 <- 1
    mul $t1, $s3, $s2   # $t1 <- $s3 * $s2

```

```

        add $t1, $t1, $s4          # $t1 += $s4
        li $t0, 4                 # $t0 <- 4
        mul $t1, $t1, $t0         # $t1 *= $t0
        add $t0, $s0, $t1         # $t0 <- $s0 + $t1
        lw $a0, ($t0)             # $a0 <- *t0
        syscall                   # syscall print_int
        li $v0, 4                 # $v0 <- 4
        la $a0, Space1            # $a1 <- &Space1
        syscall                   # syscall print_string
        addi $s4, $s4, 1          # $s4++
Loop2_Check:
        blt $s4, $s2, Loop2_Body # if $s4 < $s2 goto Loop2_Body
        li $v0, 4                 # $v0 <- 4
        la $a0, NewLine1         # $a0 <- &NewLine1
        syscall                   # syscall print_string
        addi $s3, $s3, 1          # $s3++
Loop1_Check:
        blt $s3, $s1, Loop1_Body # if $s3 < $s1 goto Loop1_Body

        lw $s4, ($sp)
        addi $sp, $sp, 4
        lw $s3, ($sp)
        addi $sp, $sp, 4
        lw $s2, ($sp)
        addi $sp, $sp, 4
        lw $s1, ($sp)
        addi $sp, $sp, 4
        lw $s0, ($sp)
        addi $sp, $sp, 4
        lw $ra, ($sp)
        addi $sp, $sp, 4

        jr $ra                    # go back to $ra

# int main()
.globl main
main:
        move $a0, $sp             # $a0 <- $sp
        addi $sp, $sp, -4         # $sp -= 4
        move $a1, $sp            # $a1 <- $sp
        jal ReadMN                # call ReadMN

        lw $s1, ($sp)             # $s1 <- *$sp
        addi $sp, $sp, 4          # $sp += 4
        lw $s0, ($sp)            # $s0 <- *$sp

        li $t0, 4                 # $t0 <- 4
        mul $t0, $t0, $s0         # $t0 *= $s0
        mul $t0, $t0, $s1         # $t0 *= $s1

        sub $sp, $sp, $t0         # $sp += $t0

        move $a0, $sp             # $a0 <- $sp
        move $a1, $s0             # $a1 <- $s0
        move $a2, $s1             # $a2 <- $s1
        jal ReadMatrix            # call ReadMatrix

        li $v0, 4                 # $v0 <- 4
        la $a0, NewLine1         # $a0 <- &NewLine1

```

<b>syscall</b>	<i># syscall print_string</i>
<b>move</b> \$a0, \$sp	<i># \$a0 &lt;- \$sp</i>
<b>move</b> \$a1, \$s0	<i># \$a1 &lt;- \$s0</i>
<b>move</b> \$a2, \$s1	<i># \$a2 &lt;- \$s1</i>
<b>jal</b> WriteMatrix	<i># call WriteMatrix</i>
<b>li</b> \$v0, 4	<i># \$v0 &lt;- 4</i>
<b>la</b> \$a0, MessageCountLinesFormat1	<i># \$a0 &lt;- &amp;MessageCountLinesFormat1</i>
<b>syscall</b>	<i># syscall print_string</i>
<b>move</b> \$a0, \$sp	<i># \$a0 &lt;- \$sp</i>
<b>move</b> \$a1, \$s0	<i># \$a1 &lt;- \$s0</i>
<b>move</b> \$a2, \$s1	<i># \$a2 &lt;- \$s1</i>
<b>jal</b> CountDifferentLines	<i># call CountDifferentLines</i>
<b>move</b> \$a0, \$v0	<i># \$a0 &lt;- \$v0</i>
<b>li</b> \$v0, 1	<i># \$v0 &lt;- 1</i>
<b>syscall</b>	<i># syscall print_int</i>
<b>li</b> \$v0, 4	<i># \$v0 &lt;- 4</i>
<b>la</b> \$a0, NewLine1	<i># \$a0 &lt;- &amp;NewLine1</i>
<b>syscall</b>	<i># syscall print_string</i>
<b>li</b> \$v0, 4	<i># \$v0 &lt;- 4</i>
<b>la</b> \$a0, MessageCountRowsFormat1	<i># \$a0 &lt;- &amp;MessageCountRowsFormat1</i>
<b>syscall</b>	<i># syscall print_string</i>
<b>move</b> \$a0, \$sp	<i># \$a0 &lt;- \$sp</i>
<b>move</b> \$a1, \$s0	<i># \$a1 &lt;- \$s0</i>
<b>move</b> \$a2, \$s1	<i># \$a2 &lt;- \$s1</i>
<b>jal</b> CountDifferentRows	<i># call CountDifferentRows</i>
<b>move</b> \$a0, \$v0	<i># \$a0 &lt;- \$v0</i>
<b>li</b> \$v0, 1	<i># \$v0 &lt;- 1</i>
<b>syscall</b>	<i># syscall print_int</i>
<b>li</b> \$v0, 4	<i># \$v0 &lt;- 4</i>
<b>la</b> \$a0, NewLine1	<i># \$a0 &lt;- &amp;NewLine1</i>
<b>syscall</b>	<i># syscall print_string</i>
<b>li</b> \$v0, 10	<i># \$v0 &lt;- 10</i>
<b>syscall</b>	<i># syscall exit</i>
<b>.data</b>	
Space1:	
<b>.ascii</b> z " "	
NewLine1:	
<b>.ascii</b> z "\n"	
InputMessagePart1:	
<b>.ascii</b> z "Введите a["	
InputMessagePart2:	
<b>.ascii</b> z "]"["	
InputMessagePart3:	
<b>.ascii</b> z "]: "	
OutputFormatMatrixPart1:	
<b>.ascii</b> z "Матрица a["	
OutputFormatMatrixPart2:	



```

        .asciiiz "]"["
OutputFormatMatrixPart3:
        .asciiiz "]:\n"

InputMNMessage1:
        .asciiiz "Введите М и N (кол-во строк и столбцов, а[M][N]): "

MessageCountLinesFormat1:
        .asciiiz "Кол-во строк, все элементы которых различны: "

MessageCountRowsFormat1:
        .asciiiz "Кол-во столбцов, все элементы которых различны: "

```

## 2.4 Реализация на ассемблере AArch64 (папка Lab\_06\_aarch64-gas)

### 2.4.1. Lab\_06\_aarch64-gas/main.s

```

// Вариант 4.
// Дана целочисленная матрица размера М x N.
// Найти количество ее строк и столбцов, все элементы которых различны.
        .text

// int main()
        .global main
main:
        stp x19, x30, [sp, #-16]!
        stp x21, x20, [sp, #-16]!

        sub sp, sp, #16                                // добавление места в стеке для двух 4-
байтных числа и выравнивание до 16 байт
        mov x20, sp                                     // x20 <- sp
        mov x1, x20                                     // x1 <- x20
        add x20, x20, #4                                 // x20 += 4
        mov x0, x20                                     // x0 <- x20
        bl ReadMN                                       // call ReadMN
        ldr w19, [x20]                                  // w19 <- *x20
        sub x20, x20, #4                                 // x20 -= 4
        ldr w21, [x20]                                  // w21 = *x20
        add sp, sp, #16                                 // sp += 16
        mov w20, w21                                   // w20 <- w21

        mul w0, w19, w20                                // w0 <- w19 * w20
        mov w1, #4                                       // w1 <- 4
        mul w0, w0, w1                                   // w0 *= 4
        bl malloc                                       // call malloc
        mov x21, x0                                     // x21 <- x0

        mov x0, x21                                     // x0 <- x21
        mov w1, w19                                       // w1 <- w19
        mov w2, w20                                       // w2 <- w20
        bl ReadMatrix                                   // call ReadMatrix

        adr x0, NewLine1                                // x0 <- &NewLine1

```

```

bl printf                                // call printf

mov x0, x21                             // x0 <- x21
mov w1, w19                             // w1 <- w19
mov w2, w20                             // w2 <- w20
bl WriteMatrix                          // call WriteMatrix

mov x0, x21                             // x0 <- x21
mov w1, w19                             // w1 <- w19
mov w2, w20                             // w2 <- w20
bl CountDifferentLines                  // call CountDifferentLines
mov w1, w0                               // w1 <- w0
adr x0, MessageCountLinesFormat1       // x0 <- &MessageCountLinesFormat1
bl printf                               // call printf

mov x0, x21                             // x0 <- x21
mov w1, w19                             // w1 <- w19
mov w2, w20                             // w2 <- w20
bl CountDifferentRows                  // call CountDifferentRows
mov w1, w0                               // w1 <- w0
adr x0, MessageCountRowsFormat1        // x0 <- &MessageCountRowsFormat1
bl printf                               // call printf

mov x0, x21                             // x0 <- x21
bl free                                 // call free

ldp x21, x20, [sp], #16
ldp x19, x30, [sp], #16

mov x0, #0                              // return 0

ret

```

MessageCountLinesFormat1:

```
.asciz "Кол-во строк, все элементы которых различны: %d\n"
```

MessageCountRowsFormat1:

```
.asciz "Кол-во столбцов, все элементы которых различны: %d\n"
```

NewLine1:

```
.asciz "\n"
```

## 2.4.2. Lab\_06\_aarch64-gas/Matrix.s

```
.text
```

```
// bool CheckAllDifferent(int* pArray, int step, int size)
```

```
.global CheckAllDifferent
```

CheckAllDifferent:

```
mul w3, w1, w2                          // w3 <- w1 * w2
```

Loop1\_Start:

```
mov w4, #0                              // w4 <- 0
```

```
b Loop1_Check                           // goto Loop1_Check
```

Loop1\_Body:

Loop2\_Start:

```

        add w5, w4, w1                // w5 <- w4 + w1
        b Loop2_Check
    Loop2_Body:
        ldr w6, [x0, x4, lsl #2]      // w6 <- x0[x4]
        ldr w7, [x0, x5, lsl #2]      // w7 <- x0[x5]
        cmp w6, w7                    // if w6 != w7
        b.ne Loop2_Continue           // goto Loop2_Continue
        mov w0, #0                    // return false
        ret
    Loop2_Continue:
        add w5, w5, w1                // w5 += w1
    Loop2_Check:
        cmp w5, w3                    // if w5 < w3
        b.lt Loop2_Body               // goto Loop2_Body
        add w4, w4, w1                // w4 += w1
    Loop1_Check:
        cmp w4, w3                    // if w4 < w3
        b.lt Loop1_Body               // goto Loop1_Body

    mov w0, #1                        // return true
    ret

```

*// int CountDifferentLines(int\* pMatrix, int m, int n)*

**.global** CountDifferentLines

CountDifferentLines:

```

    stp x19, x30, [sp, #-16]!
    stp x23, x20, [sp, #-16]!
    stp x25, x24, [sp, #-16]!
    stp xzr, x26, [sp, #-16]!

```

```

    mov x23, x0                      // x23 <- x0
    mov w24, w1                      // w24 <- w1
    sxtw x25, w2                     // x25 <- w2

```

```

    mov x26, #0                      // x26 <- 0

```

```

    mov w19, #0                      // w19 <- 0

```

*// for (w20 = 0, w20 < w24, w20++)*

Loop3\_Start:

```

    mov w20, #0                      // w20 <- 0
    b Loop3_Check                    // goto Loop3_Check

```

Loop3\_Body:

```

    mov x0, #4                      // x0 <- 4
    madd x0, x26, x0, x23            // x0 <- x26 * 4 + x23
    mov w1, #1                      // w1 <- 1
    mov x2, x25                     // x2 <- x25
    bl CheckAllDifferent             // call CheckAllDifferent
    cmp w0, #0                      // if x0 == false
    b.eq Loop3_Continue              // goto Loop3_Continue
    add w19, w19, #1                 // w19++

```

Loop3\_Continue:

```

    add x26, x26, x25                // x26 += x25
    add w20, w20, #1                 // x20++

```

Loop3\_Check:

```

    cmp w20, w24                    // if x20 < x24
    b.lt Loop3_Body                  // goto Loop3_Body

```

```

    mov w0, w19                      // w0 <- w19

```

```

ldp xzr, x26, [sp], #16
ldp x25, x24, [sp], #16
ldp x23, x20, [sp], #16
ldp x19, x30, [sp], #16
ret                                     // return w0 = w19

// int CountDifferentRows(int* pMatrix, int m, int n)
.global CountDifferentRows
CountDifferentRows:
    stp x19, x30, [sp, #-16]!
    stp x23, x20, [sp, #-16]!
    stp x25, x24, [sp, #-16]!

    mov x23, x0                       // x23 <- x0
    mov w24, w1                       // w24 <- w1
    mov w25, w2                       // w25 <- w2

    mov w19, #0                      // w19 <- 0

    // for (w20 = 0, w20 < w25, w20++)
Loop4_Start:
    mov w20, #0                      // w20 <- 0
    b Loop4_Check                    // goto Loop4_Check
Loop4_Body:
    mov x0, #4                       // x0 <- 4
    madd x0, x20, x0, x23            // x0 <- x20 * 4 + x23
    mov w1, w25                      // w1 <- w25
    mov w2, w24                      // w2 <- w24
    bl CheckAllDifferent             // call CheckAllDifferent
    cmp w0, #0                      // if x0 == false
    b.eq Loop4_Continue              // goto Loop4_Continue
    add w19, w19, #1                 // w19++
Loop4_Continue:
    add w20, w20, #1                 // w20++
Loop4_Check:
    cmp w20, w25                    // if w20 < w25
    b.lt Loop4_Body                 // goto Loop4_Body

    mov w0, w19                     // w0 <- w19

    ldp x25, x24, [sp], #16
    ldp x23, x20, [sp], #16
    ldp x19, x30, [sp], #16
    ret                             // return w0 = w19

```

### 2.4.3. Lab\_06\_aarch64-gas/MatrixIO.s

```

.text

// void ReadMN(int* pM, int* pN)
.global ReadMN
ReadMN:
    stp x19, x30, [sp, #-16]!
    stp xzr, x20, [sp, #-16]!

```

```

mov x19, x0          // x19 <- x0
mov x20, x1          // x20 <- x1

adr x0, InputMNMessage1 // x0 <- &InputMNMessage1
bl printf           // call printf

adr x0, InputMNFormat1 // x0 <- &InputMNFormat1
mov x1, x19          // x1 <- x19
mov x2, x20           // x2 <- x20
bl scanf             // call scanf

ldp xzr, x20, [sp], #16
ldp x19, x30, [sp], #16
ret

// void ReadMatrix(int* pMatrix, int m, int n)
.global ReadMatrix
ReadMatrix:
stp x19, x30, [sp, #-16]!
stp x21, x20, [sp, #-16]!
stp x23, x22, [sp, #-16]!

mov x19, x0          // x19 <- x0
mov w20, w1          // w20 <- w1
mov w21, w2          // w21 <- w2

// for (w22 <- 0, w22 < x20, w22++)
Loop3_Start:
mov w22, #0          // w22 <- 0
b Loop3_Check        // goto Loop3_Check
Loop3_Body:
// for (w23 <- 0, w23 < x21, w23++)
Loop4_Start:
mov w23, #0          // w23 <- 0
b Loop4_Check        // goto Loop4_Check
Loop4_Body:
adr x0, InputMessageFormat1 // x0 <- &InputMessageFormat1
mov w1, w22          // w1 <- w22
mov w2, w23          // w2 <- w23
bl printf           // call printf
adr x0, InputFormat1 // x0 <- &InputFormat1
madd w1, w22, w21, w23 // w1 <- w22 * w21 + w23
mov w2, #4           // w2 <- 4
mul w1, w1, w2        // w1 *= w2
add x1, x19, x1       // x1 += x19
bl scanf             // call scanf
add w23, w23, #1      // w23++
Loop4_Check:
cmp w23, w21         // if w23 < w21
b.lt Loop4_Body      // goto Loop4_Body
add w22, w22, #1      // w22++
Loop3_Check:
cmp w22, w20         // if w22 < w20
b.lt Loop3_Body      // goto Loop3_Body

ldp x23, x22, [sp], #16
ldp x21, x20, [sp], #16
ldp x19, x30, [sp], #16

```

**ret**

```
// void WriteMatrix(int* pMatrix, int m, int n)
.global WriteMatrix
WriteMatrix:
    stp x19, x30, [sp, #-16]!
    stp x21, x20, [sp, #-16]!
    stp x23, x22, [sp, #-16]!

    mov x19, x0                // x19 <- x0
    mov w20, w1                // w20 <- w1
    mov w21, w2                // w21 <- w2

    adr x0, OutputFormatMatrix1 // x0 <- &OutputFormatMatrix1
    bl printf                  // call printf

    // for (w22 <- 0, w22 < x20, w22++)
Loop1_Start:
    mov w22, #0                // w22 <- 0
    b Loop1_Check              // goto Loop1_Check
Loop1_Body:
    // for (w23 <- 0, w23 < x21, w23++)
    Loop2_Start:
        mov w23, #0            // w23 <- 0
        b Loop2_Check          // goto Loop2_Check
    Loop2_Body:
        adr x0, OutputFormat1  // x0 <- &OutputFormat1
        madd w1, w22, w21, w23 // w1 <- w22 * w21 + w23
        ldr w1, [x19, x1, lsl#2] // w1 <- x19[x1]
        bl printf              // call printf
        add w23, w23, #1        // w23++
    Loop2_Check:
        cmp w23, w21            // if w23 < w21
        b.lt Loop2_Body         // goto Loop2_Body
        adr x0, NewLine          // x0 <- &NewLine
        bl printf              // call printf
        add w22, w22, #1        // w22++
    Loop1_Check:
        cmp w22, w20            // if w22 < w20
        b.lt Loop1_Body         // goto Loop1_Body

    ldp x23, x22, [sp], #16
    ldp x21, x20, [sp], #16
    ldp x19, x30, [sp], #16
    ret
```

```
OutputFormatMatrix1:
    .asciz "Матрица a[%d][%d]:\n"
```

```
OutputFormat1:
    .asciz "%d "
```

```
NewLine:
    .asciz "\n"
```

```
InputMessageFormat1:
    .asciz "Введите a[%d][%d]: "
```

```
InputFormat1:  
    .asciz "%d"
```

```
InputMNMessage1:  
    .asciz "Введите М и N (кол-во строк и столбцов, а[М][N]): "
```

```
InputMNFormat1:  
    .asciz "%d%d"
```

## 2.5 Реализация на Си (папка Lab\_06\_C)

### 2.5.1. Lab\_06\_C/main.c

```
// \file  
// \brief Вариант 4.  
// \details Дана целочисленная матрица размера М х N.  
// Найти количество ее строк и столбцов, все элементы которых различны.  
  
#include <stdio.h>  
#include <malloc.h>  
  
#include "Matrix.h"  
#include "MatrixIO.h"  
  
int main()  
{  
    int m;  
    int n;  
  
    ReadMN(&m, &n);  
  
    int* pMatrix = (int*) malloc(m * n * sizeof (int));  
  
    ReadMatrix(pMatrix, m, n);  
  
    printf("\n");  
  
    WriteMatrix(pMatrix, m, n);  
  
    printf("Кол-во строк, все элементы которых различны: %d\n",  
        CountDifferentLines(pMatrix, m, n));  
    printf("Кол-во столбцов, все элементы которых различны: %d\n",  
        CountDifferentRows(pMatrix, m, n));  
  
    free(pMatrix);  
  
    return 0;  
}
```

### 2.5.2. Lab\_06\_C/Matrix.c

```

/// \file
/// \brief Реализация функций из Matrix.h
/// \details Реализация функций из Matrix.h.

#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>

#include "Matrix.h"

bool CheckAllDifferent(int* pArray, int step, int size)
{
    int max = size * step;
    for (int i = 0; i < max; i += step)
    {
        for (int j = i + step; j < max; j += step)
        {
            if (pArray[i] == pArray[j])
            {
                return false;
            }
        }
    }
    return true;
}

int CountDifferentLines(int* pMatrix, int m, int n)
{
    int c = 0;
    for (int i = 0; i < m; i++)
    {
        bool check = CheckAllDifferent(pMatrix + i * n, 1, n);
        if (check)
        {
            c++;
        }
    }
    return c;
}

int CountDifferentRows(int* pMatrix, int m, int n)
{
    int c = 0;
    for (int i = 0; i < n; i++)
    {
        bool check = CheckAllDifferent(pMatrix + i, n, m);
        if (check)
        {
            c++;
        }
    }
    return c;
}

```



### 2.5.3. Lab\_06\_C/MatrixIO.c

```
/// \file
/// \brief Реализация функций из MatrixIO.h
/// \details Реализация функций из MatrixIO.h.

#include <stdio.h>

#include "MatrixIO.h"

void ReadMN(int* pM, int* pN)
{
    printf("Введите M и N (кол-во строк и столбцов, a[M][N]): ");
    scanf("%d%d", pM, pN);
}

void ReadMatrix(int* pMatrix, int m, int n)
{
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("Введите a[%d][%d]: ", i, j);
            scanf("%d", &pMatrix[i * n + j]);
        }
    }
}

void WriteMatrix(int* pMatrix, int m, int n)
{
    printf("Матрица a[%d][%d]:\n", m, n);
    for (int i = 0; i < m; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%d ", pMatrix[i * n + j]);
        }
        printf("\n");
    }
}
```

## 3 Тестовые примеры работы программ

### 3.1 Пример сборки и работы под x86\_32

#### 3.1.1. Сборка и запуск основной программы на ассемблере (WSL Ubuntu 18.04)

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Main-i386
make -C Lab_06_i386-gas
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
as -c --32 main.s -o main.o
as -c --32 Matrix.s -o Matrix.o
as -c --32 MatrixIO.s -o MatrixIO.o
gcc main.o Matrix.o MatrixIO.o -m32 -o Lab_06_i386-gas
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ file ./Lab_06_i386-gas/Lab_06_i386-gas
./Lab_06_i386-gas/Lab_06_i386-gas: ELF 32-bit LSB shared object, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-, for GNU/Linux 3.2.0, BuildID[sha1]=cd86d94004b552c1198becfb4024200efe768b58, not stripped
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ ./Lab_06_i386-gas/Lab_06_i386-gas
-bash: ./Lab_06_i386-gas/Lab_06_i386-gas: cannot execute binary file: Exec format error
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 2 – Сборка и неудачная попытка запуска

Из-за особенностей Windows Subsystem for Linux, невозможен запуск 32-битного приложения. Обойти это можно через qemu в user-режиме, или запустив на «настоящем» линуксе.

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ qemu-i386 ./Lab_06_i386-gas/Lab_06_i386-gas
Введите M и N (кол-во строк и столбцов, a[M][N]): 2
3
Введите a[0][0]: 1
Введите a[0][1]: 2
Введите a[0][2]: 3
Введите a[1][0]: 1
Введите a[1][1]: 2
Введите a[1][2]: 3
Матрица a[2][3]:
1 2 3
1 2 3
Кол-во строк, все элементы которых различны: 2
Кол-во столбцов, все элементы которых различны: 0
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 3 – Запуск через qemu-i386

### 3.1.2. Сборка и запуск основной программы на ассемблере (Ubuntu 16.04)

```
vladislav@MySandbox: ~/Projects/SystemProgramming/0x0E/Lab_06
vladislav@MySandbox:~/Projects/SystemProgramming/0x0E/Lab_06$ make Main-i386
make -C Lab_06_i386-gas
make[1]: Entering directory '/home/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
as -c --32 main.s -o main.o
as -c --32 Matrix.s -o Matrix.o
as -c --32 MatrixIO.s -o MatrixIO.o
gcc main.o Matrix.o MatrixIO.o -m32 -o Lab_06_i386-gas
make[1]: Leaving directory '/home/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
vladislav@MySandbox:~/Projects/SystemProgramming/0x0E/Lab_06$ ./Lab_06_i386-gas/Lab_06_i386-gas
Введите M и N (кол-во строк и столбцов, a[M][N]): 3
2
Введите a[0][0]: 1
Введите a[0][1]: 1
Введите a[1][0]: 1
Введите a[1][1]: 2
Введите a[2][0]: 1
Введите a[2][1]: 1

Матрица a[3][2]:
1 1
1 2
1 1
Кол-во строк, все элементы которых различны: 1
Кол-во столбцов, все элементы которых различны: 0
vladislav@MySandbox:~/Projects/SystemProgramming/0x0E/Lab_06$ _
```

Рисунок 4 – Сборка и удачный запуск на «настоящем» линуксе

### 3.1.3. Сборка и запуск модульных тестов (WSL Ubuntu 18.04)

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06
vladislav@DESKTOP-ODR692H:/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Tests-i386
make -C Lab_06_i386-gas
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
make -C Lab_06_Tests CC=gcc CFLAGS="-O0 -Wall -std=gnu99 -m32" LDFLAGS="-m32" LCUNIT="-L./CUnit/i386 -lcunit" PROJOBJ="..../Lab_06_i386-gas/Matrix.o ../Lab_06_i386-gas/MatrixIO.o"
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
gcc -c -I.. -I.. -O0 -Wall -std=gnu99 -m32 main.c -o main.o
gcc -c -I.. -I.. -O0 -Wall -std=gnu99 -m32 MatrixTests.c -o MatrixTests.o
gcc -c -I.. -I.. -O0 -Wall -std=gnu99 -m32 Suite.c -o Suite.o
gcc main.o MatrixTests.o Suite.o ../Lab_06_i386-gas/Matrix.o ../Lab_06_i386-gas/MatrixIO.o -L./CUnit/i386 -lcunit -m32 -o Lab_06_Tests
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
vladislav@DESKTOP-ODR692H:/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ qemu-i386 ./Lab_06_Tests/Lab_06_Tests

CUnit - A unit testing framework for C - Version 3.1.1-cunity
http://cunit.sourceforge.net/

----- Начат Test_CountLineRowAndWrite_5x4_0to19
Матрица a[5][4]:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
Кол-во строк, все элементы которых различны: 5
Кол-во столбцов, все элементы которых различны: 4

---- Закончен Test_CountLineRowAndWrite_5x4_0to19, прошло 0.000000 секунд
```

Рисунок 5 – Сборка и запуск тестов (начало)

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06
Матрица a[3][6]:
5 3 2 3 9 8
9 1 1 6 4 2
3 0 8 5 7 2
Кол-во строк, все элементы которых различны: 1
Кол-во столбцов, все элементы которых различны: 5
Матрица a[3][6]:
2 0 7 2 3 8
6 9 3 3 5 3
5 0 8 9 6 9
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 5

---- Закончен Example_CountLineRowAndWrite_3x6_RandomThrice, прошло 0.000000 секунд
---- Начат Benchmark_CountLinesRows_10x15x20x500
---- Закончен Benchmark_CountLinesRows_10x15x20x500, прошло 0.093750 секунд
---- Начат Benchmark_CountLinesRows_500x1000x1
---- Закончен Benchmark_CountLinesRows_500x1000x1, прошло 1.750000 секунд

Run Summary      -      Run      Failed      Inactive      Skipped
Suites           :           1           0           0           0
Asserts          :          40           0          n/a          n/a
Tests            :           14           0           0           0

Elapsed Time: 1.859(s)

vladislav@DESKTOP-ODR692H:/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 6 – Запуск тестов (конец)

## 3.2 Пример сборки и работы под AArch64

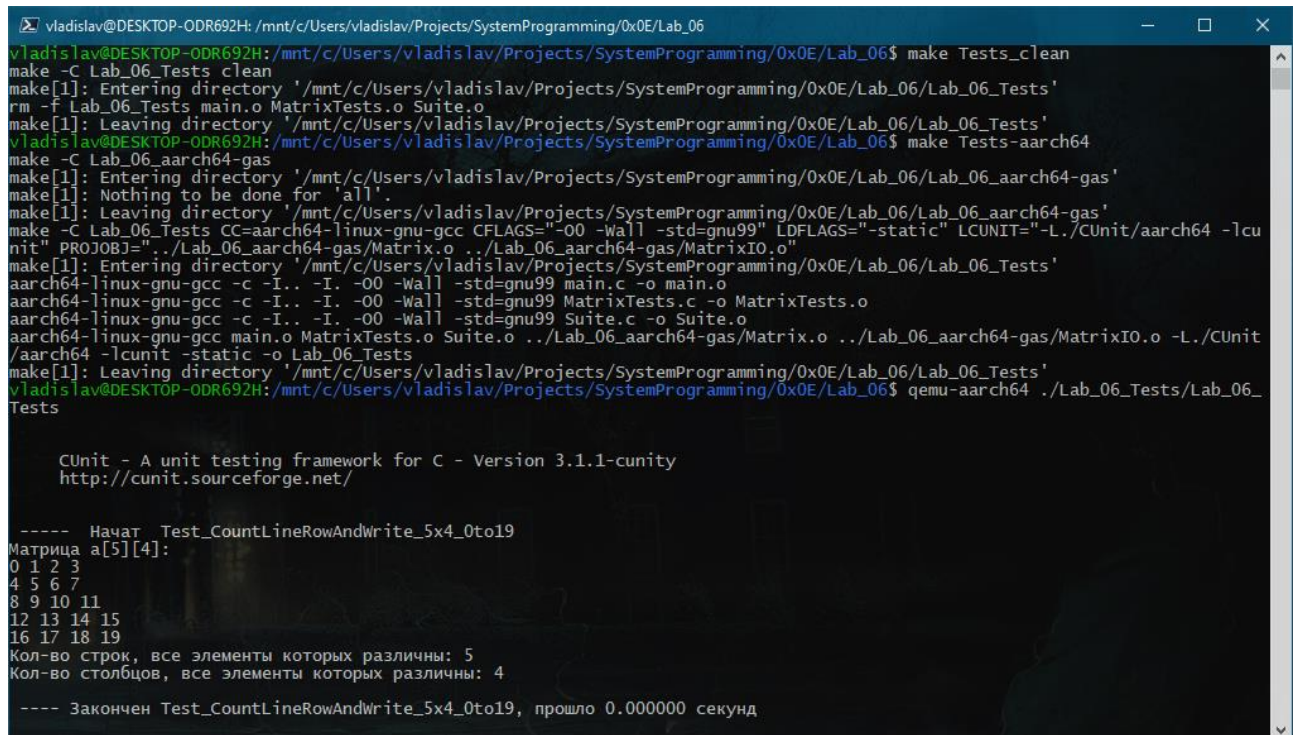
### 3.2.1. Сборка и запуск основной программы на ассемблере (WSL Ubuntu 18.04)

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06
vladislav@DESKTOP-ODR692H:/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Main-aarch64
make -C Lab_06_aarch64-gas
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
aarch64-linux-gnu-as -c main.s -o main.o
aarch64-linux-gnu-as -c Matrix.s -o Matrix.o
aarch64-linux-gnu-as -c MatrixIO.s -o MatrixIO.o
aarch64-linux-gnu-gcc main.o Matrix.o MatrixIO.o -static -o Lab_06_aarch64-gas
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
vladislav@DESKTOP-ODR692H:/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ qemu-aarch64 ./Lab_06_aarch64-gas/Lab_06_aarch64-gas
Введите M и N (кол-во строк и столбцов, a[M][N]): 1
3
Введите a[0][0]: 123
Введите a[0][1]: 324
Введите a[0][2]: 123
Матрица a[1][3]:
123 324 123
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 3
vladislav@DESKTOP-ODR692H:/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 7 – Сборка и запуск основной программы



### 3.2.2. Очистка тестов от тестов для x86\_32, сборка и запуск тестов (WSL Ubuntu 18.04)

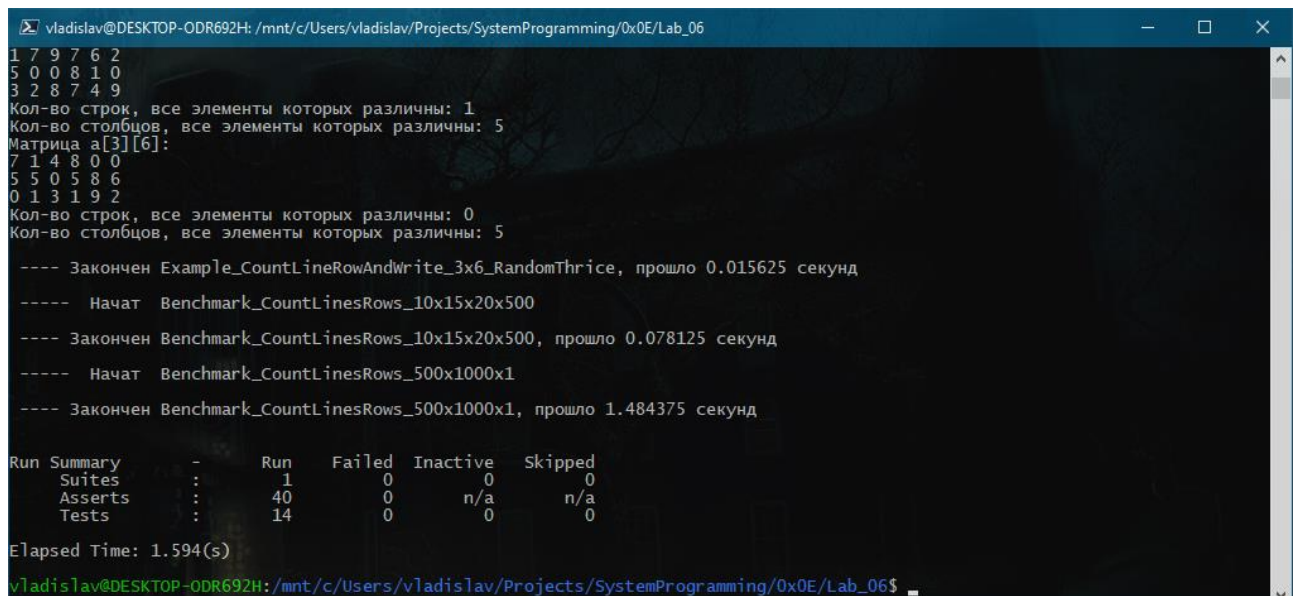


```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Tests_clean
make -C Lab_06_Tests clean
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
rm -f Lab_06_Tests main.o MatrixTests.o Suite.o
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Tests-aarch64
make -C Lab_06_aarch64-gas
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
make -C Lab_06_Tests CC=aarch64-linux-gnu-gcc CFLAGS="-O0 -Wall -std=gnu99" LDFLAGS="-static" LCUNIT="-L./CUnit/aarch64 -lcunit" PROJOBJ="-" ./Lab_06_aarch64-gas/Matrix.o ./Lab_06_aarch64-gas/MatrixIO.o
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
aarch64-linux-gnu-gcc -c -I. -I. -O0 -Wall -std=gnu99 main.c -o main.o
aarch64-linux-gnu-gcc -c -I. -I. -O0 -Wall -std=gnu99 MatrixTests.c -o MatrixTests.o
aarch64-linux-gnu-gcc -c -I. -I. -O0 -Wall -std=gnu99 Suite.c -o Suite.o
aarch64-linux-gnu-gcc main.o MatrixTests.o Suite.o ./Lab_06_aarch64-gas/Matrix.o ./Lab_06_aarch64-gas/MatrixIO.o -L./CUnit/aarch64 -lcunit -static -o Lab_06_Tests
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ qemu-aarch64 ./Lab_06_Tests/Lab_06_Tests

CUnit - A unit testing framework for C - Version 3.1.1-cunity
http://cunit.sourceforge.net/

----- Начат Test_CountLineRowAndWrite_5x4_0to19
Матрица a[5][4]:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
Кол-во строк, все элементы которых различны: 5
Кол-во столбцов, все элементы которых различны: 4
---- Закончен Test_CountLineRowAndWrite_5x4_0to19, прошло 0.000000 секунд
```

Рисунок 8 – Очистка, сборка и запуск тестов (начало)



```
1 7 9 7 6 2
5 0 0 8 1 0
3 2 8 7 4 9
Кол-во строк, все элементы которых различны: 1
Кол-во столбцов, все элементы которых различны: 5
Матрица a[3][6]:
7 1 4 8 0 0
5 5 0 5 8 6
0 1 3 1 9 2
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 5

---- Закончен Example_CountLineRowAndWrite_3x6_RandomThrice, прошло 0.015625 секунд
----- Начат Benchmark_CountLinesRows_10x15x20x500
---- Закончен Benchmark_CountLinesRows_10x15x20x500, прошло 0.078125 секунд
----- Начат Benchmark_CountLinesRows_500x1000x1
---- Закончен Benchmark_CountLinesRows_500x1000x1, прошло 1.484375 секунд

Run Summary      -      Run   Failed Inactive Skipped
Suites           :        1         0         0         0
Asserts          :       40         0        n/a        n/a
Tests            :       14         0         0         0

Elapsed Time: 1.594(s)
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 9 – Запуск тестов (конец)

### 3.2.3. Сборка, запуск основной программы на ассемблере и тестов (Android)

```
20:21:39 25%
$ make Main-aarch64-android
make -C Lab_06_aarch64-gas AS=as CC=clang LDFLAGS=
make[1]: Entering directory '/data/data/com.termux/files/home/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
as -c main.s -o main.o
as -c Matrix.s -o Matrix.o
as -c MatrixIO.s -o MatrixIO.o
clang main.o Matrix.o MatrixIO.o -o Lab_06_aarch64-gas
make[1]: Leaving directory '/data/data/com.termux/files/home/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
$ ./Lab_06_aarch64-gas/Lab_06_aarch64-gas
Введите M и N (кол-во строк и столбцов, a[M][N]): 2
3
Введите a[0][0]: 4
Введите a[0][1]: 4
Введите a[0][2]: 4
Введите a[1][0]: 4
Введите a[1][1]: 4
Введите a[1][2]: 4

Матрица a[2][3]:
4 4 4
4 4 4
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 0
$ make Tests-aarch64-android
make -C Lab_06_aarch64-gas AS=as CC=clang LDFLAGS=
make[1]: Entering directory '/data/data/com.termux/files/home/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
make[1]: Nothing to be done for 'all'.
make[1]: Leaving directory '/data/data/com.termux/files/home/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
make -C Lab_06_Tests CC=clang CFLAGS="-O0 -Wall -std=gnu99" LDFLAGS= LCUNIT="-L./CUnit/android -lcunit" PROJOBJ="./Lab_06_aarch64-gas/Matrix.o ../Lab_06_aarch64-gas/MatrixIO.o"
make[1]: Entering directory '/data/data/com.termux/files/home/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
clang -c -I. -I. -O0 -Wall -std=gnu99 main.c -o main.o
clang -c -I. -I. -O0 -Wall -std=gnu99 MatrixTests.c -o MatrixTests.o
clang -c -I. -I. -O0 -Wall -std=gnu99 Suite.c -o Suite.o
clang main.o MatrixTests.o Suite.o ../Lab_06_aarch64-gas/Matrix.o ../Lab_06_aarch64-gas/MatrixIO.o -L./CUnit/android -lcunit -o Lab_06_Tests
make[1]: Leaving directory '/data/data/com.termux/files/home/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
$ ./Lab_06_Tests/Lab_06_Tests

CUnit - A unit testing framework for C - Version 3.1.1-cunity
http://cunit.sourceforge.net/

----- Начат Test_CountLineRowAndWrite_5x4_0to19
Матрица a[5][4]:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
Кол-во строк, все элементы которых различны: 5
Кол-во столбцов, все элементы которых различны: 4

---- Закончен Test_CountLineRowAndWrite_5x4_0to19, прошло 0.000121 сек
УНД

----- Начат Test_CountLineRowAndWrite_5x4_All0
Матрица a[5][4]:
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

ESC CTRL ALT — ↓ ↑
```

Рисунок 10 – Сборка основной программы, запуск основной программы, сборка тестов и запуск тестов (начало)

```

20:21:48 🎵 📺 📶 4G+ 25%
Матрица a[0][0]:
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 0

---- Закончен Test_CountLineRowAndWrite_0x0_1, прошло 0.000019 секунд

----- Начат Test_CountLineRowAndWrite_3x3_Custom1
Матрица a[3][3]:
1 -23 12
23 -23 123
123 123 1
Кол-во строк, все элементы которых различны: 2
Кол-во столбцов, все элементы которых различны: 2

---- Закончен Test_CountLineRowAndWrite_3x3_Custom1, прошло 0.000045 секунд

----- Начат Test_CountLineRowAndWrite_3x3_Custom2
Матрица a[3][3]:
1 -23 12
23 -23 123
123 123 123
Кол-во строк, все элементы которых различны: 2
Кол-во столбцов, все элементы которых различны: 1

---- Закончен Test_CountLineRowAndWrite_3x3_Custom2, прошло 0.000044 секунд

----- Начат Example_CountLineRowAndWrite_3x6_RandomThrice
Матрица a[3][6]:
4 7 8 0 9 8
1 3 4 5 4 6
1 6 5 6 2 1
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 5
Матрица a[3][6]:
9 5 3 7 1 1
0 0 4 5 4 8
3 0 7 3 1 6
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 4
Матрица a[3][6]:
1 4 0 8 1 4
4 2 3 2 8 5
5 9 2 8 7 5
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 4

---- Закончен Example_CountLineRowAndWrite_3x6_RandomThrice, прошло 0.000693 секунд

----- Начат Benchmark_CountLinesRows_10x15x20x500

---- Закончен Benchmark_CountLinesRows_10x15x20x500, прошло 0.052106 секунд

----- Начат Benchmark_CountLinesRows_500x1000x1

---- Закончен Benchmark_CountLinesRows_500x1000x1, прошло 0.454618 секунд

Run Summary      -      Run      Failed      Inactive      Skipped
Suites           :           1           0           0           0
Asserts          :          40           0          n/a          n/a
Tests            :          14           0           0           0

Elapsed Time: 0.508(s)
$ █
ESC  ⌨  CTRL  ALT  —  ↓  ↑

```

Рисунок 11 – Запуск тестов (конец)

### 3.3 Запуск основной программы под MIPS32 в симуляторе SPIM

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ spim -f ./Lab_06_mips-spim/Lab_06_mips-spim.s
SPIM Version 8.0 of January 8, 2010
Copyright 1990-2010, James R. Larus.
All Rights Reserved.
See the file README for a full copyright notice.
Loaded: /usr/lib/spim/exceptions.s
Введите M и N (кол-во строк и столбцов, a[M][N]): 2
3
Введите a[0][0]: 1
Введите a[0][1]: 2
Введите a[0][2]: 3
Введите a[1][0]: -214
Введите a[1][1]: 2
Введите a[1][2]: -214
Матрица a[2][3]:
1 2 3
-214 2 -214
Кол-во строк, все элементы которых различны: 1
Кол-во столбцов, все элементы которых различны: 2
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 12 – Запуск основной программы

## 3.4 Пример сборки и работы реализации на Си

### 3.4.1. Сборка и запуск (WSL Ubuntu 18.04)

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Main-C
make -C Lab_06_C
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_C'
gcc -c -I.. -O0 -Wall -std=gnu99 main.c -o main.o
gcc -c -I.. -O0 -Wall -std=gnu99 Matrix.c -o Matrix.o
gcc -c -I.. -O0 -Wall -std=gnu99 MatrixIO.c -o MatrixIO.o
gcc main.o Matrix.o MatrixIO.o -o Lab_06_C
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_C'
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ ./Lab_06_C/Lab_06_C
Введите M и N (кол-во строк и столбцов, a[M][N]): 2
6
Введите a[0][0]: 1
Введите a[0][1]: 2
Введите a[0][2]: 3
Введите a[0][3]: 5
Введите a[0][4]: 21
Введите a[0][5]: 3
Введите a[1][0]: 2
Введите a[1][1]: 2
Введите a[1][2]: 3
Введите a[1][3]: 42
Введите a[1][4]: 1
Введите a[1][5]: 2
Матрица a[2][6]:
1 2 3 5 21 3
2 2 3 42 1 2
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 4
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 13 – Сборка и запуск основной программы

## 3.5 Сборка и запуск тестов (WSL Ubuntu 18.04)



```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make clean
make -C Lab_06_i386-gas clean
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
rm -f Lab_06_i386-gas main.o Matrix.o MatrixIO.o
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_i386-gas'
make -C Lab_06_aarch64-gas clean
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
rm -f Lab_06_aarch64-gas main.o Matrix.o MatrixIO.o
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_aarch64-gas'
make -C Lab_06_C clean
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_C'
rm -f Lab_06_C main.o Matrix.o MatrixIO.o
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_C'
make -C Lab_06_Tests clean
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
rm -f Lab_06_Tests main.o MatrixTests.o Suite.o
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ make Tests-C
make -C Lab_06_C
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_C'
gcc -c -I.. -O0 -Wall -std=gnu99 main.c -o main.o
gcc -c -I.. -O0 -Wall -std=gnu99 Matrix.c -o Matrix.o
gcc -c -I.. -O0 -Wall -std=gnu99 MatrixIO.c -o MatrixIO.o
gcc main.o Matrix.o MatrixIO.o -o Lab_06_C
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_C'
make -C Lab_06_Tests
make[1]: Entering directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
gcc -c -I.. -I. -O0 -Wall -std=gnu99 main.c -o main.o
gcc -c -I.. -I. -O0 -Wall -std=gnu99 MatrixTests.c -o MatrixTests.o
gcc -c -I.. -I. -O0 -Wall -std=gnu99 Suite.c -o Suite.o
gcc main.o MatrixTests.o Suite.o ../Lab_06_C/MatrixIO.o -lcunit -no-pie -o Lab_06_Tests
make[1]: Leaving directory '/mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06/Lab_06_Tests'
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ ./Lab_06_Tests/Lab_06_Tests

CUnit - A unit testing framework for C - Version 3.1.1-cunity
http://cunit.sourceforge.net/

----- Начат Test_CountLineRowAndWrite_5x4_0to19
Матрица a[5][4]:
0 1 2 3
4 5 6 7
8 9 10 11
12 13 14 15
16 17 18 19
Кол-во строк, все элементы которых различны: 5
Кол-во столбцов, все элементы которых различны: 4
```

Рисунок 14 – Полная очистка, сборка тестов и запуск (начало)

```
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$ ./Lab_06_Tests/Lab_06_Tests
1 7 7 7 3 3
Кол-во строк, все элементы которых различны: 0
Кол-во столбцов, все элементы которых различны: 5

---- Закончен Example_CountLineRowAndWrite_3x6_RandomThrice, прошло 0.015625 секунд
----- Начат Benchmark_CountLinesRows_10x15x20x500
---- Закончен Benchmark_CountLinesRows_10x15x20x500, прошло 0.046875 секунд
----- Начат Benchmark_CountLinesRows_500x1000x1
---- Закончен Benchmark_CountLinesRows_500x1000x1, прошло 0.890625 секунд

Run Summary      -      Run      Failed      Inactive      Skipped
Suites           :         1          0          0          0
Asserts          :        40         n/a         n/a         n/a
Tests            :        14          0          0          0

Elapsed Time: 0.953(s)
vladislav@DESKTOP-ODR692H: /mnt/c/Users/vladislav/Projects/SystemProgramming/0x0E/Lab_06$
```

Рисунок 15 – Запуск тестов (конец)