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 \_\_\_\_\_\_\_\_\_ А.С. Кузнецов

подпись, дата инициалы, фамилия

Студент КИ18-17/1б 031831229 \_\_\_\_\_\_\_\_\_ В.А. Прекель

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2019

Содержание

[Содержание 2](#_Toc27508652)

[1 Цель работы с постановкой задачи 4](#_Toc27508653)

[1.1 Цель работы 4](#_Toc27508654)

[1.2 Задача работы 4](#_Toc27508655)

[1.3 Описание и пояснение к работе 4](#_Toc27508656)

[2 Комментированный исходный код программ. 6](#_Toc27508657)

[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](#_Toc27508658)

[2.1.1. Matrix.h 6](#_Toc27508659)

[2.1.2. MatrixIO.h 7](#_Toc27508660)

[2.2 Реализация на ассемблере x86\_32 (папка Lab\_06\_i386-gas) 7](#_Toc27508661)

[2.2.1. Lab\_06\_i386-gas/main.s 7](#_Toc27508662)

[2.2.2. Lab\_06\_i386-gas/Matrix.s 9](#_Toc27508663)

[2.2.3. Lab\_06\_i386-gas/MatrixIO.s 13](#_Toc27508664)

[2.3 Реализация на ассемблере MIPS32 (папка Lab\_06\_mips-spim) 17](#_Toc27508665)

[2.3.1. Lab\_06\_mips-spim/Lab\_06\_mips-spim.s 17](#_Toc27508666)

[2.4 Реализация на ассемблере AArch64 (папка Lab\_06\_aarch64-gas) 25](#_Toc27508667)

[2.4.1. Lab\_06\_aarch64-gas/main.s 25](#_Toc27508668)

[2.4.2. Lab\_06\_aarch64-gas/Matrix.s 26](#_Toc27508669)

[2.4.3. Lab\_06\_aarch64-gas/MatrixIO.s 28](#_Toc27508670)

[2.5 Реализация на Си (папка Lab\_06\_C) 31](#_Toc27508671)

[2.5.1. Lab\_06\_C/main.c 31](#_Toc27508672)

[2.5.2. Lab\_06\_C/Matrix.c 31](#_Toc27508673)

[2.5.3. Lab\_06\_C/MatrixIO.c 33](#_Toc27508674)

[3 Тестовые примеры работы программ 33](#_Toc27508675)

[3.1 Пример сборки и работы под x86\_32 33](#_Toc27508676)

[3.1.1. Сборка и запуск основной программы на ассемблере (WSL Ubuntu 18.04) 33](#_Toc27508677)

[3.1.2. Сборка и запуск основной программы на ассемблере (Ubuntu 16.04) 34](#_Toc27508678)

[3.1.3. Сборка и запуск модульных тестов (WSL Ubuntu 18.04) 35](#_Toc27508679)

[3.2 Пример сборки и работы под AArch64 36](#_Toc27508680)

[3.2.1. Сборка и запуск основной программы на ассемблере (WSL Ubuntu 18.04) 36](#_Toc27508681)

[3.2.2. Очистка тестов от тестов для x86\_32, сборка и запуск тестов (WSL Ubuntu 18.04) 37](#_Toc27508682)

[3.2.3. Сборка, запуск основной программы на ассемблере и тестов (Android) 38](#_Toc27508683)

[3.3 Запуск основной программы под MIPS32 в симуляторе SPIM 39](#_Toc27508684)

[3.4 Пример сборки и работы реализации на Си 40](#_Toc27508685)

[3.4.1. Сборка и запуск (WSL Ubuntu 18.04) 40](#_Toc27508686)

[3.5 Сборка и запуск тестов (WSL Ubuntu 18.04) 40](#_Toc27508687)

# Цель работы с постановкой задачи

## Цель работы

Разработка программ на языках ассемблера.

## 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 x N. Найти количество ее строк и столбцов, все элементы которых различны.

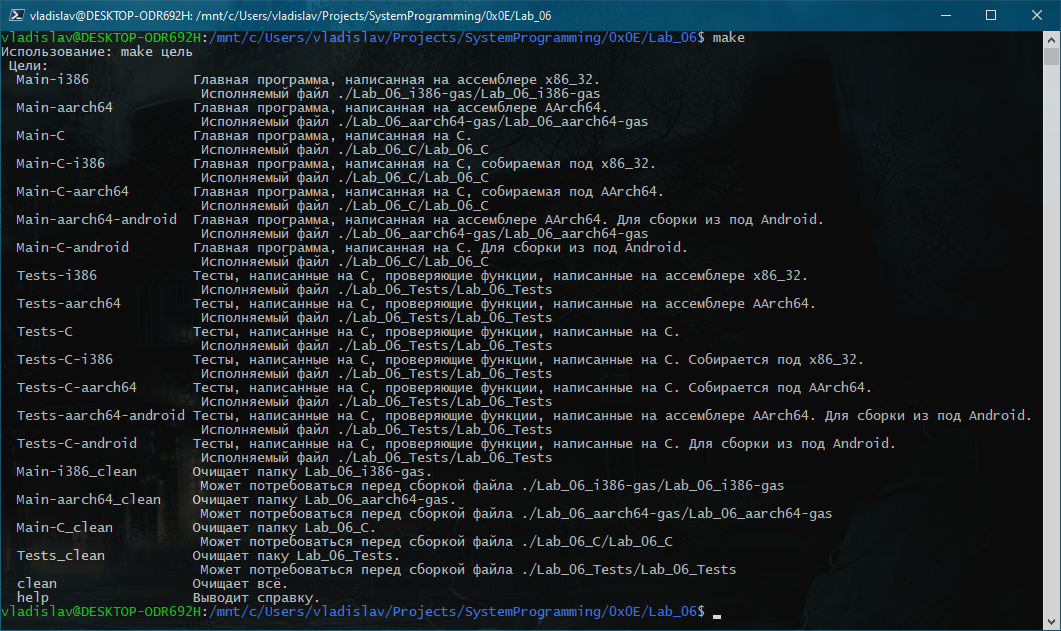
## Описание и пояснение к работе

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.



1. Вывод справки

# Комментированный исходный код программ.

## 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)

### 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, хрянящейнся 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*

### 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 Указетель на матрицу.  
/// \param m Кол-во строк.  
/// \param n Колв-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*

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

### 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"

### 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  
 **jl** 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  
 **jl** 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**

### 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  
 **jl** Loop4\_Body  
 **movl** -4(**%**ebp), **%**eax  
 **addl** $1, **%**eax  
 **movl %**eax, -4(**%**ebp)  
 Loop3\_Check:  
 **movl** -4(**%**ebp), **%**eax  
 **cmpl** 12(**%**ebp), **%**eax  
 **jl** 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 M 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"

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

### Lab\_06\_mips-spim/Lab\_06\_mips-spim.s

*# Вариант 4.   
# Дана целочисленная матрица размера M x N.   
# Найти 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 *# $s0 <- $a0* **move** $s1, $a1 *# $s1 <- $a1* **move** $s2, $a2 *# $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* **syscall** *# syscall print\_string* **move** $a0, $sp *# $a0 <- $sp* **move** $a1, $s0 *# $a1 <- $s0* **move** $a2, $s1 *# $a2 <- $s1* **jal** WriteMatrix *# call WriteMatrix* **li** $v0, 4 *# $v0 <- 4* **la** $a0, MessageCountLinesFormat1 *# $a0 <- &MessageCountLinesFormat1* **syscall** *# syscall print\_string* **move** $a0, $sp *# $a0 <- $sp* **move** $a1, $s0 *# $a1 <- $s0* **move** $a2, $s1 *# $a2 <- $s1* **jal** CountDifferentLines *# call CountDifferentLines* **move** $a0, $v0 *# $a0 <- $v0* **li** $v0, 1 *# $v0 <- 1* **syscall** *# syscall print\_int* **li** $v0, 4 *# $v0 <- 4* **la** $a0, NewLine1 *# $a0 <- &NewLine1* **syscall** *# syscall print\_string* **li** $v0, 4 *# $v0 <- 4* **la** $a0, MessageCountRowsFormat1 *# $a0 <- &MessageCountRowsFormat1* **syscall** *# syscall print\_string* **move** $a0, $sp *# $a0 <- $sp* **move** $a1, $s0 *# $a1 <- $s0* **move** $a2, $s1 *# $a2 <- $s1* **jal** CountDifferentRows *# call CountDifferentRows* **move** $a0, $v0 *# $a0 <- $v0* **li** $v0, 1 *# $v0 <- 1* **syscall** *# syscall print\_int* **li** $v0, 4 *# $v0 <- 4* **la** $a0, NewLine1 *# $a0 <- &NewLine1* **syscall** *# syscall print\_string* **li** $v0, 10 *# $v0 <- 10* **syscall** *# syscall exit* **.data**Space1:   
 **.asciiz** " "  
  
NewLine1:  
 **.asciiz** "\n"  
  
InputMessagePart1:  
 **.asciiz** "Введите a["  
  
InputMessagePart2:  
 **.asciiz** "]["  
  
InputMessagePart3:  
 **.asciiz** "]: "  
  
OutputFormatMatrixPart1:  
 **.asciiz** "Матрица a["  
  
OutputFormatMatrixPart2:  
 **.asciiz** "]["  
  
OutputFormatMatrixPart3:  
 **.asciiz** "]:\n"  
  
InputMNMessage1:  
 **.asciiz** "Введите M и N (кол-во строк и столбцов, a[M][N]): "  
  
MessageCountLinesFormat1:  
 **.asciiz** "Кол-во строк, все элементы которых различны: "  
   
MessageCountRowsFormat1:  
 **.asciiz** "Кол-во столбцов, все элементы которых различны: "

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

### Lab\_06\_aarch64-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()* **.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"

### 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*

### 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 M и N (кол-во строк и столбцов, a[M][N]): "  
  
InputMNFormat1:  
 **.asciz** "%d%d"

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

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

*// \file  
// \brief Вариант 4.  
// \details Дана целочисленная матрица размера M x 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;  
}

### 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;  
}

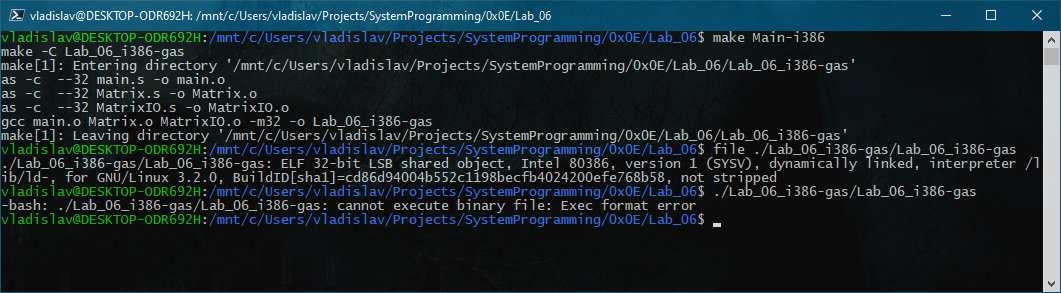
### Lab\_06\_C/MatrixIO.c

*/// \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"**);  
 }  
}

# Тестовые примеры работы программ

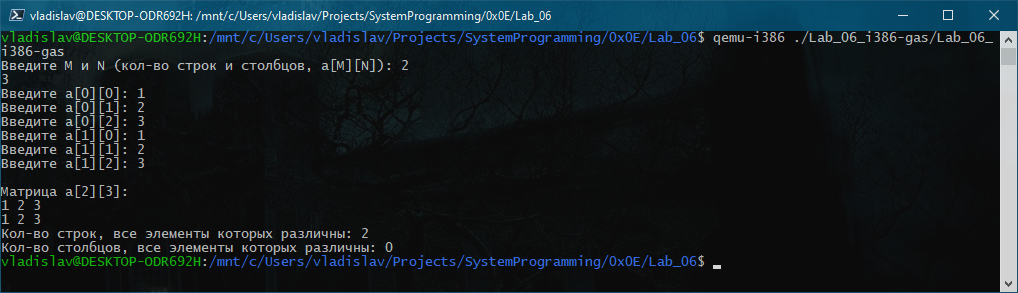
## Пример сборки и работы под x86\_32

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



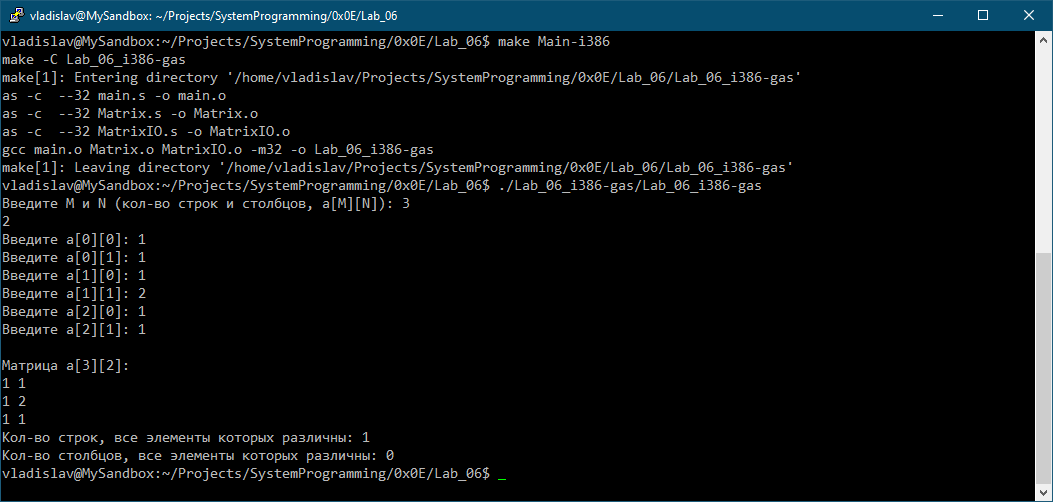
1. Сборка и неудачная попытка запуска

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.



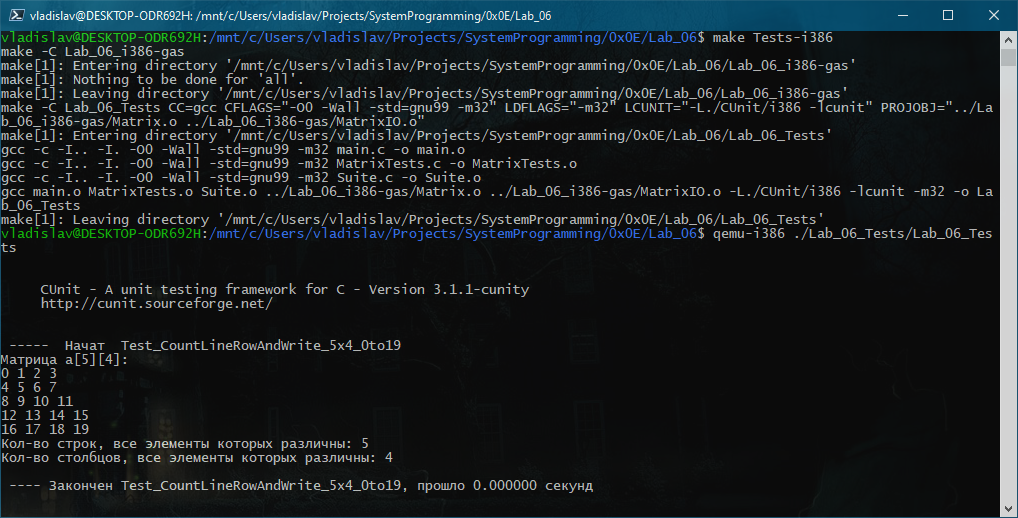
1. Запуск через qemu-i386

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

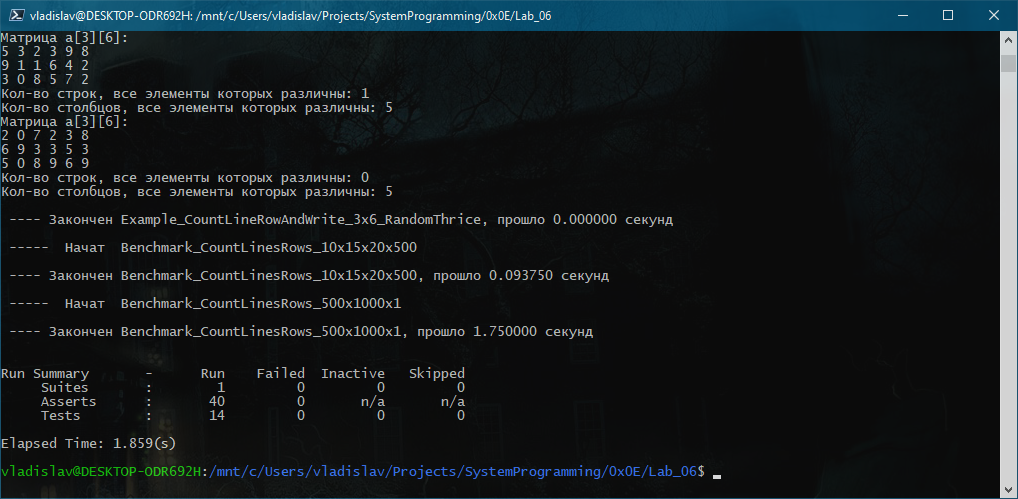


1. Сборка и удачный запуск на «настоящем» линуксе

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



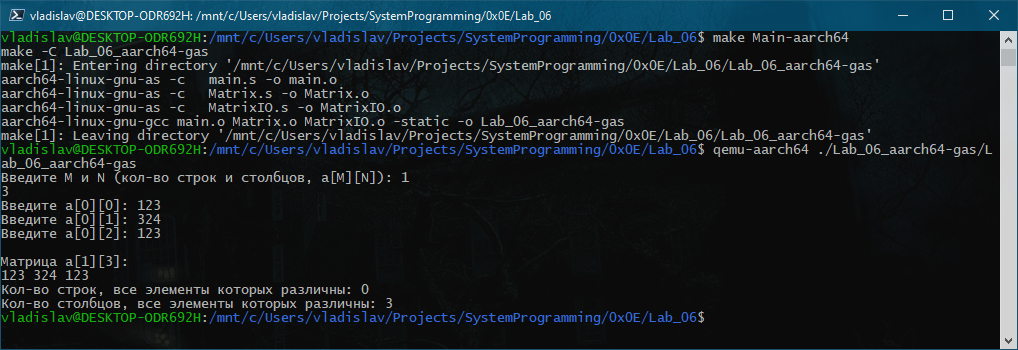
1. Сборка и запуск тестов (начало)



1. Запуск тестов (конец)

## Пример сборки и работы под AArch64

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

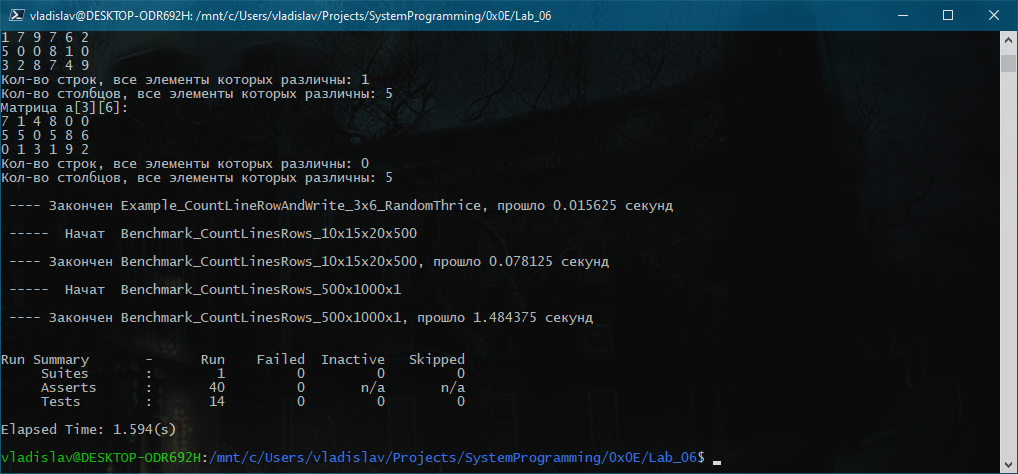


1. Сборка и запуск основной программы

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

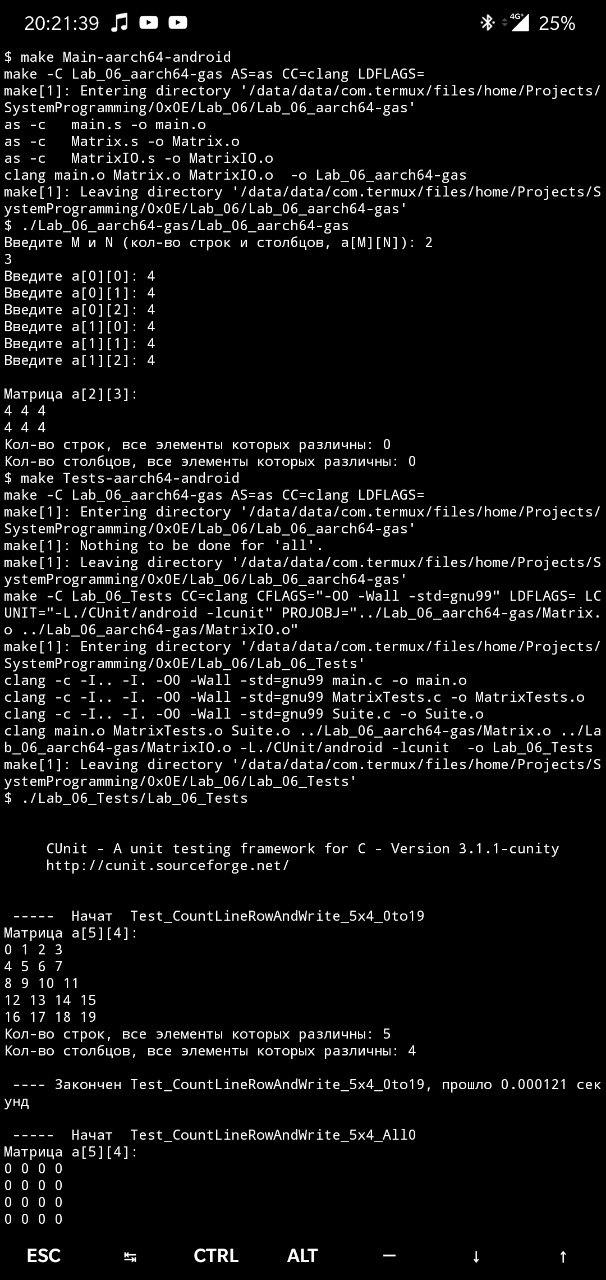


1. Очистка, сборка и запуск тестов (начало)

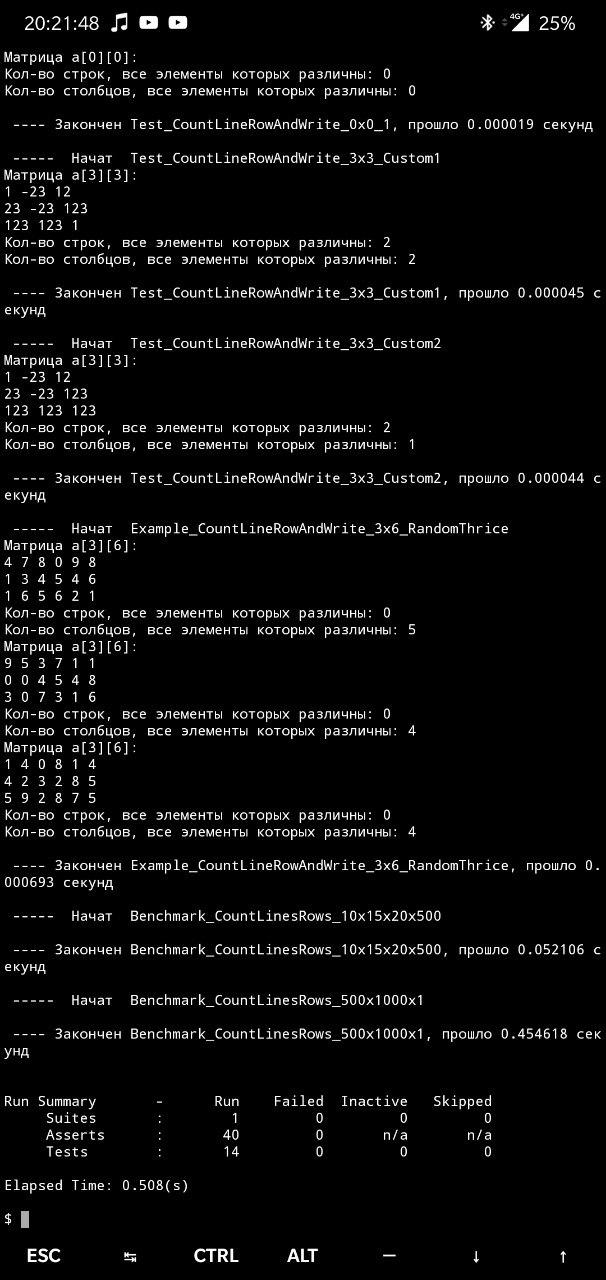


1. Запуск тестов (конец)

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



1. Сборка основной программы, запуск основной программы, сборка тестов и запуск тестов (начало)



1. Запуск тестов (конец)

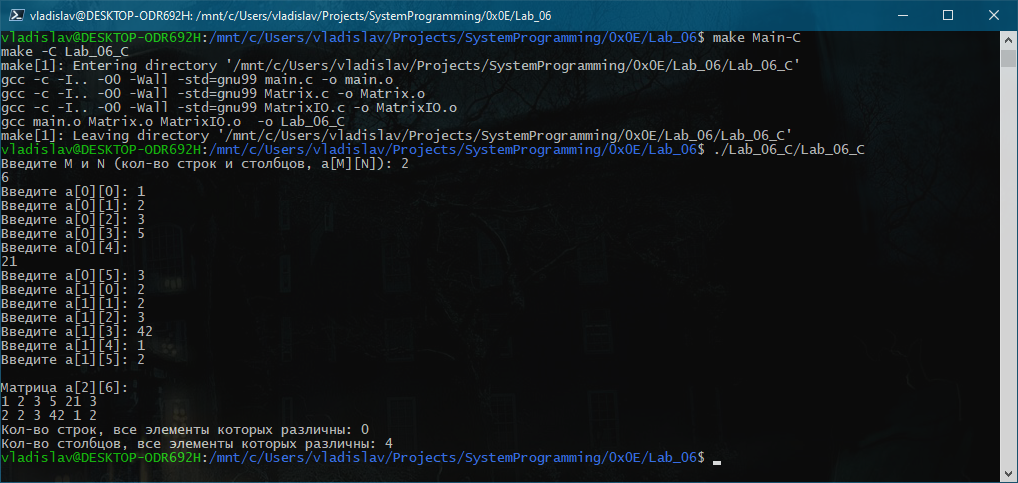
## Запуск основной программы под MIPS32 в симуляторе SPIM



1. Запуск основной программы

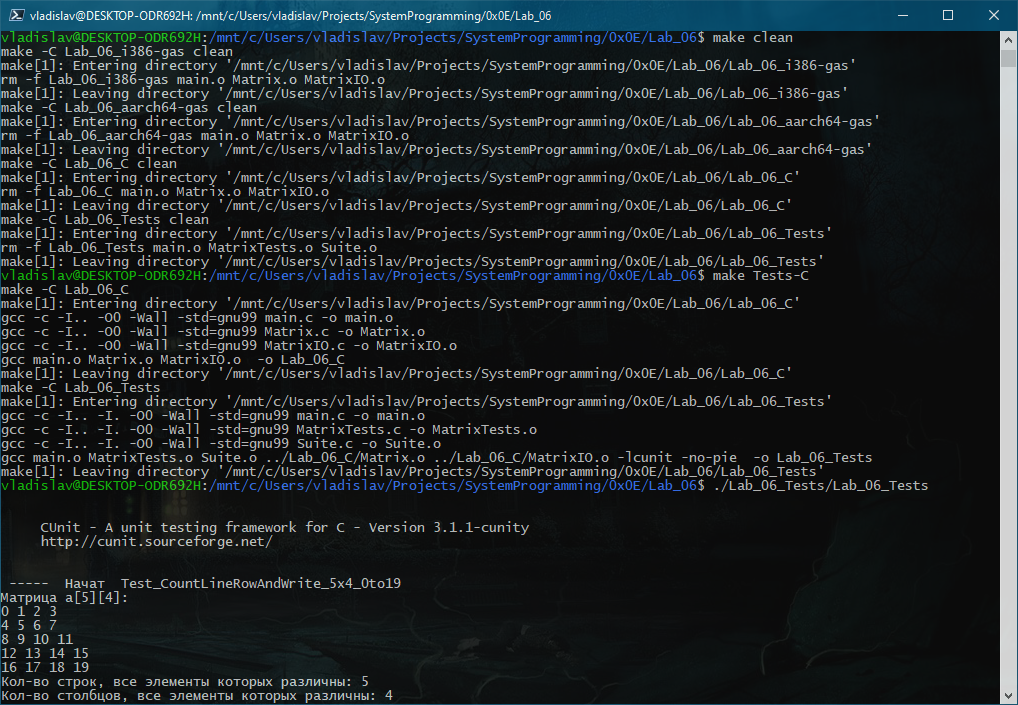
## Пример сборки и работы реализации на Си

### Сборка и запуск (WSL Ubuntu 18.04)

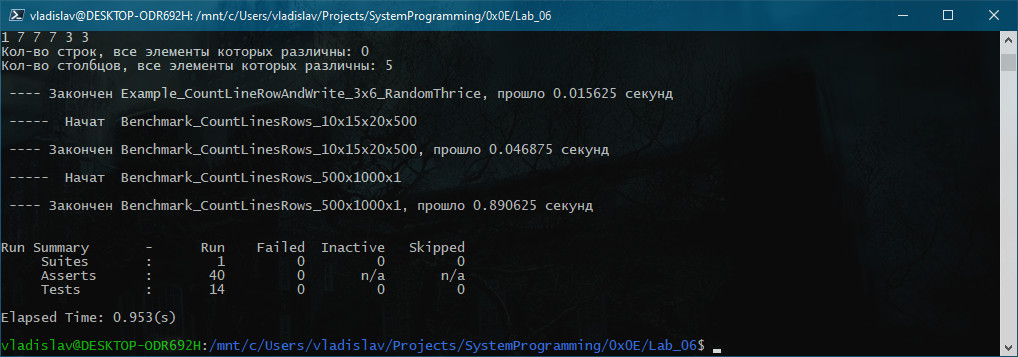


1. Сборка и запуск основной программы

## Сборка и запуск тестов (WSL Ubuntu 18.04)



1. Полная очистка, сборка тестов и запуск (начало)



1. Запуск тестов (конец)