

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе №6
по дисциплине «Организация ЭВМ и систем»
Тема: Организация связи Ассемблера с ЯВУ на примере
программы построения частотного распределение попаданий
псевдослучайных целых чисел в заданные интервалы.

Студент гр. 1381

Мелькумянц Д.А.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Получить практические навыки программирования на языке Ассемблера.

Разработать программу на ЯВУ с использованием языка Ассемблера.

Задание.

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

Далее должен вызываться ассемблерный модуль для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. В общем случае интервалы разбиения диапазона изменения псевдослучайных чисел могут иметь различную длину.

Результирующий массив частотного распределения чисел по интервалам, сформированный на ассемблерном уровне, возвращается в программу, реализованную на ЯВУ, и затем сохраняется в файле и выводится на экран средствами ЯВУ.

Исходные данные.

1. Длина массива псевдослучайных целых чисел - NumRanDat ($\leq 16K$, $K=1024$)
2. Диапазон изменения массива псевдослучайных целых чисел $[X_{\min}, X_{\max}]$, значения могут быть биполярные;
3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (≤ 24)
4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу $[X_{\min}, X_{\max}]$).

Результаты:

1. Текстовый файл, строка которого содержит:

- номер интервала,

- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал.

Количество строк равно числу интервалов разбиения.

2. График, отражающий распределение чисел по интервалам.

(необязательный результат)

Подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля, сразу формирующего требуемое распределение и возвращающего его в главную программу, написанную на ЯВУ.

Выполнение работы.

На языке C++ реализовано считывание начальных данных. Левые границы заносятся в массив `LgrInt`, а генерируемые числа добавляются в массив `arr`. Отдельно создается массив, который будет хранить результат работы.

В ассемблерный модуль в процедуру `FUNC` передаются указатель на массив сгенерированных чисел и его размер, указатель на массив левых границ интервалов и его размер, указатель на массив, хранящих результат работы. В процедуре для каждого числа находится интервал, в который он попадает, и результат записывается в результирующий массив.

После этого результат работы выводится в консоль и записывается в файл `out.txt`.

Рис. 1 - Проверка работы программы.

```

Введите количество псевдослучайных чисел
20
Введите минимальное значение
5
Введите максимальное значение
95
Введите количество интервалов
4
Введите левые границы
5 15 25 35
Индекс интервала      Интервал левой границы      Число чисел в интервале
1                      25                          0
2                      15                          0
3                      5                           4
4                      35                          12

```

Вывод.

В ходе выполнения данной лабораторной работы была написана программа на языке Ассемблера, которая строит частотное распределение попаданий псевдослучайных чисел в заданные интервалы. В результате выполнения лабораторной работы были получены практические навыки программирования на языке Ассемблер.

Приложение А

Исходный код программ

Название файла: Source.cpp

```
#include <iostream>
#include <fstream>
#include <random>

extern "C" void FUNC(int* array, int array_size, int* LGrInt, int NInt, int*
result_array);

int main() {
    system("chcp 1251 > nul");
    setlocale(LC_CTYPE, "rus");
    std::ofstream file("out.txt");
    int NumRanDat;

    std::cout << "Введите количество псевдослучайных чисел\n";
    std::cin >> NumRanDat;

    int Xmax, Xmin;
    std::cout << "Введите минимальное значени\n";
    std::cin >> Xmin;
    std::cout << "Введите максимальное значение\n";
    std::cin >> Xmax;

    if (Xmax < Xmin) {
        std::cout << "Неверные значения\n";
        return 0;
    }

    int NInt;
    std::cout << "Введите количетсво интервалов\n";
    std::cin >> NInt;

    if (NInt <= 0) {
        std::cout << "Неверное значения\n";
        return 0;
    }

    int* LGrInt = new int[NInt];
    std::cout << "Введите левые границы\n";
    for (int i = 0; i < NInt; i++) {
```

```

        std::cin >> LGrInt[i];
    }

    for (int i = 0; i < NInt - 1; i++) {
        for (int j = 0; j < NInt; j++) {
            if (LGrInt[j] < LGrInt[i])
                std::swap(LGrInt[j], LGrInt[i]);
        }
    }

    std::random_device rand;
    std::mt19937 gen(rand());
    std::uniform_int_distribution<> dis(Xmin, Xmax);
    int* arr = new int[NumRanDat];
    for (int i = 0; i < NumRanDat; i++) {
        arr[i] = dis(gen);
    }

    for (int i = 0; i < NumRanDat; i++) {
        file << arr[i] << ' ';
    }
    file << '\n';

    int* result_arr = new int[NInt];
    for (int i = 0; i < NInt; i++) {
        result_arr[i] = 0;
    }

    FUNC(arr, NumRanDat, LGrInt, NInt, result_arr);

    std::cout << "Индекс интервала \t Интервал левой границы \t Число
чисел в интервале\n";
    file << "Индекс интервала \t Интервал левой границы \t Число чисел в
интервале\n";

    for (int i = 0; i < NInt; i++) {
        std::cout << "\t" << i + 1 << "\t\t" << LGrInt[i] << "\t\t\t" <<
result_arr[i] << '\n';
        file << "\t" << i + 1 << "\t\t" << LGrInt[i] << "\t\t\t" <<
result_arr[i] << '\n';
    }
    delete[] LGrInt;
    delete[] arr;
    delete[] result_arr;

```

```
}
```

Название файла: module.asm

.586

.MODEL FLAT, C

.CODE

FUNC PROC C array:dword, array_size:dword, left_borders:dword,
interval_amount:dword, result_array:dword

push ecx

push esi

push edi

push eax

push ebx

mov ecx, array_size

mov esi, array

mov edi, left_borders

mov eax, 0

l1:

mov ebx, 0

borders:

cmp ebx, interval_amount

jge borders_exit

push eax

mov eax, [esi+4*eax]

cmp eax, [edi+4*ebx]

pop eax

j1 borders_exit

inc ebx

jmp borders

borders_exit:

dec ebx

cmp ebx, -1

je skip

mov edi, result_array

push eax

mov eax, [edi+4*ebx]

inc eax

mov [edi+4*ebx], eax

pop eax

```
        mov edi, left_borders
        skip:
        inc eax
loop l1

pop ebx
pop eax
pop edi
pop esi
pop ecx
ret
FUNC ENDP
    END
```