

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №6**  
**по дисциплине «Организация ЭВМ и систем»**  
**Тема: Организация связи Ассемблера с ЯВУ.**

Студент гр. 9382

Савельев И.С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

### **Цель работы.**

Организовать связи Ассемблера с ЯВУ на примере программы построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

### **Задание.**

На языке высокого уровня (Pascal или C) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND\_GEN (при его отсутствии программу датчика получить у преподавателя). Далее должен вызываться ассемблерный модуль(модули) 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 ( $\leq 24$ )
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)

### **Ход выполнения.**

В качестве ЯВУ использовался C++. С его помощью считывались все необходимые данные, после чего вызывался ассемблерный модуль, который обрабатывал входной массив и возвращал готовый результат в C++.

### **Тестирование.**

Входные данные	Выходные данные
Введите длину массива: 10 Введите нижний интервал: 1 Введите верхний интервал: 10 Введите количество интервалов: 2 Введите 1 нижних границ интервалов: 5	Набор случайных чисел: 2 6 2 6 5 4 7 5 4 9  Номер интервала    Левая граница    Количество чисел 1                      1                      6 2                      5                      4
Введите длину массива: 50 Введите нижний интервал: 1 Введите верхний интервал: 50 Введите количество интервалов: 3 Введите 2 нижних границ интервалов: 0 Введенная граница 0 не входит в заданные промежутки! Введите еще раз: 10 25	Набор случайных чисел: 30 21 38 16 27 16 45 9 18 39 25 20 26 2 20 19 12 26 32 35 1 38 2 26 45 27 22 19 48 9 3 28 49 49 48 24 20 30 9 42 26 10 35 36 35 38 15 34 3 22  Номер интервала    Левая граница    Количество чисел 1                      1                      9 2                      10                    15 3                      25                    26

### **Вывод.**

Была реализована программа организующая связь ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

## Приложение А. Исходный код программы.

Файл main.cpp

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

extern "C" void INTERVAL_SORTING(int64_t* LGrInt, int64_t*
borderult, int64_t* array, int64_t NInt);

int main() {
    int64_t NInt = 0;
    int64_t Xmin = 0;
    int64_t Xmax = 0;
    int64_t count = 0;
    int64_t border;

    std::cout << "Введите длину массива: ";
    std::cin >> NInt;
    // Проверка длины массива
    while (NInt > 16384) {
        std::cout << "Длина больше допустимой,
введите заново: ";
        std::cin >> NInt;
    }

    std::cout << "Введите нижний интервал: ";
    std::cin >> Xmin;

    std::cout << "Введите верхний интервал: ";
    std::cin >> Xmax;
    // Проверка верхнего интервала
    while (Xmax <= Xmin) {
        std::cout << "Введен не корректный верхний
интервал, введите еще раз: " << '\n';
        std::cin >> Xmax;
    }

    std::cout << "Введите количество
интервалов: ";
    std::cin >> count;
    // Проверка интервалов
    while (count > 24) {
        std::cout << "Введено не корректное число,
введите еще раз: ";
        std::cin >> count;
    }
}
```

```

        int64_t *LGrInt = new int64_t[count];
        std::cout << "Введите " << count - 1 << " нижних
границ интервалов: ";
        // Считывание нижних границ
        for (int64_t i = 0; i < count - 1; i++) {
            std::cin >> LGrInt[i];
            while (LGrInt[i] > Xmax || LGrInt[i] < Xmin) {
                std::cout << "Введенная граница " <<
LGrInt[i] << " не входит в заданные промежутки!
Введите еще раз: ";
                std::cin >> LGrInt[i];
            }
        }

        LGrInt[count - 1] = Xmax;

        int64_t *array = new int64_t[NInt];
        // Генерация псевдослучайных чисел
        for (int64_t i = 0; i < NInt; i++) {
            array[i] = Xmin + rand() % (Xmax - Xmin);
        }
        // Обнуляем массив ответов
        int64_t *borderult = new int64_t[count];
        for (int64_t i = 0; i < count; i++) {
            borderult[i] = 0;
        }
        // Вызов ассемблерного модуля
        INTERVAL_SORTING(LGrInt, borderult, array, NInt);
        // Запись в файл и вывод на экран
        std::ofstream out_file("borderult.txt");
        std::cout << "Набор случайных чисел: ";
        out_file << "Набор случайных чисел: ";
        for (int64_t i = 0; i < NInt; i++) {
            std::cout << array[i] << " ";
            out_file << array[i] << " ";
        }
        out_file << "\n";
        std::cout << "\n";
        std::cout << "\nНомер интервала\tЛевая
граница\tКоличество чисел\n";
        out_file << "\nНомер интервала\tЛевая
граница\tКоличество чисел\n";

        for (int64_t i = 0; i < count; i++) {
            if(i != 0) {
                border = LGrInt[i - 1];
            }
            else {
                border = Xmin;
            }
        }

```

```

        out_file << "    " << i+1 << "\t\t" << border <<
"\t\t\t" << borderult[i] << "\n";
        std::cout << "    " << i+1 << "\t" << border <<
"\t\t" << borderult[i] << "\n";
    }
    // О с в о б о ж д е н и е   п а м я т и
    delete borderult;
    delete array;
    delete LGrInt;

}

```

Файл `assem.s`

```

.intel_syntax noprefix
.global INTERVAL_SORTING
.text

INTERVAL_SORTING: # rdi : LGrInt,   rsi : result ,   rdx : array,
rcx : NInt

    WokWork:
        mov rax, [rdx]           # в rax лежит текущий
элемент
        push rdx                 # сохраняем указатель на
текущий элемент
        mov rdx, 0               # обнуляем указатель

    Index_case:
        mov rbx, rdx             # в rbx лежит текущий индекс
из LGrInt
        shl rbx, 3               # этот индекс умножаем на 8,
т.е. каждый элемент по 8 байт
        cmp rax, [rdi + rbx]     # сравниваем текущий
элемент массива с текущей границей
        jg search_case           # если элемент
массива больше границы
        jmp write_case

    search_case:
        inc rdx                  # для перехода к
следующему индексу массива границ
        jmp Index_case

    write_case:
        add rbx, rsi             # rbx указывает на индекс
LGrInt
        mov rax, [rbx]           # rax хранит индекс LGrInt
который надо ++

```

```

        inc rax                                #    увеличиваем
индекс на один
        mov [rbx], rax;                        #    возвращаем
обратно
        pop rdx                                #
восстанавливаем указатель на array
        add rdx, 8                             #    перемещаем
указатель на следующий элемент массива
чисел
        loop WokWork                          #    в цикле проходим по
всем элементам массива

        ret

```