

**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 на примере**  
**программы построения частотного распределение попаданий**  
**псевдослучайных целых чисел в заданные интервалы.**

Студентка гр. 1303

Куклина Ю.Н.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2022

## **Цель работы.**

Рассмотреть способ организации связи ассемблера с ЯВУ на примере связи с языком программирования С. Разработать программу, выполняющую подсчет попаданий псевдослучайных чисел в заданные интервалы.

## **Задание.**

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND\_GEN (при его отсутствии получить у преподавателя). Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

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.

## **Выполнение работы:**

Вариант 17.

Программа написана в двух модулях. Первый реализован на языке Си, где происходит считывание входных данных и запись результата. Второй – на языке Ассемблера, где происходит обработка данных.

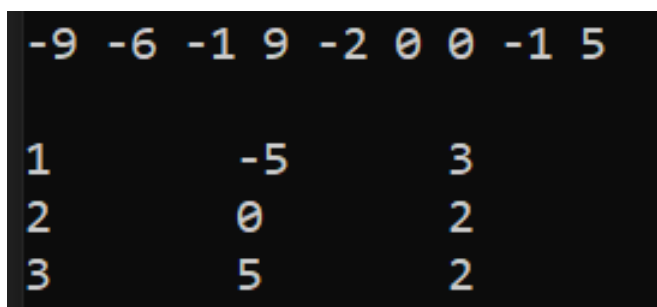
В начале программы в модуле Си происходит считывание входных данных и необходимые проверки на их корректность, в случае ошибки –

программа завершается.

После вызывается процедура *f* из ассемблерного модуля, в который передаётся указатель на результирующий массив, исходный массив, количество чисел, количество левых границ и массив левых границ. В ней загружается число из исходной строки в регистр *eax*, которое после обрабатывается в процедуре *find* - процедура, находящая номер интервала, в которое входит данное число. В процедуре *find*, начиная с последней левой границы обходятся все интервалы, и если число становится большим или равным текущей левой границе, значит, это число принадлежит этому интервалу — цикл поиска номера кончается, номер записывается в регистрах, процедура завершается; если такой интервал не был найден, в *eax* записывается 0. Далее, основная процедура проверяет, нашёлся ли такой интервал, и если да, то увеличивает необходимую позицию в результирующем массиве на 1. Затем считывается следующее число пока *ecx* не будет равен нулю (в *ecx* изначально хранится количество элементов в исходном массиве сгенерированных чисел).

Исходный код программы представлен в приложении А.

На рисунке 1 представлен результат работы программы.



-9 -6 -1 9 -2 0 0 -1 5								
1			-5				3	
2			0				2	
3			5				2	

Рисунок1.

### **Вывод.**

Рассмотрен способ организации связи Ассемблера с ЯВУ. Разработана программа построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы.

## Приложение А

### 1. Файл Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

extern void f(int *result_array, int *source_array, int
*borders_array,
            int number, int borders_number);

int main() {
    srand(time(NULL));

    int n = 0;
    int x_min = 0;
    int x_max = 0;

    int n_int = 0;

    printf("Введите количество чисел, левую границу, правую границу и
число "
           "левых границ\n");
    scanf("%d %d %d %d", &n, &x_min, &x_max, &n_int);

    if (n <= 0 || n > 16 * 1024) {
        printf("Некорректное количество чисел\n");
        return 1;
    } else if (x_min >= x_max) {
        printf("Некорректные границы\n");
        return 1;
    } else if (n_int <= 0 || n_int > 24) {
        printf("Некорректные границы\n");
        return 1;
    }

    int *n_arr = malloc(n * sizeof(int));
    int *int_arr = malloc(n_int * sizeof(int));

    printf("Введите левые границы\n");
    char c;
    for (int i = 0; i < n_int; ++i) {
        scanf("%d%c", &int_arr[i], &c);
        if ((int_arr[i] < x_min || int_arr[i] > x_max) ||
            (i > 0 && int_arr[i] <= int_arr[i - 1])) {
            printf("Некорректная левая граница\n");
            free(n_arr);
            free(int_arr);
            return 1;
        }
    }

    int rand_max = x_max - x_min + 1;
    for (int i = 0; i < n; ++i) {
        n_arr[i] = x_min + rand() % rand_max;
    }
}
```

```

int *res_arr = calloc(n_int, sizeof(int));
f(res_arr, n_arr, int_arr, n, n_int);

FILE *f = fopen("results.txt", "w");
for (int i = 0; i < n; ++i) {
    fprintf(f, "%d ", n_arr[i]);
}
fputs("\n\n", f);
for (int i = 0; i < n_int; ++i) {
    fprintf(f, "%d    %d    %d\n", i + 1, int_arr[i], res_arr[i]);
}

fclose(f);

free(res_arr);
free(n_arr);
free(int_arr);
return 0;
};

```

## 2. Файл Lib.s

```
.global f
```

```
find:
```

```
find_loop:
```

```
    cmp    eax, [rdx + rcx * 4 - 4]
    jge    find_end
    loop   find_loop
```

```
find_end:
```

```
    mov    rax, rcx
```

```
    ret
```

```
f:
```

```
    push   rax
```

```
f_loop:
```

```
    lodsd
```

```
    push   rcx
```

```
    mov    rcx, r8
```

```
    call   find
```

```
    pop    rcx
```

```
    cmp    rax, 0
```

```
    je     continue_loop
```

```
    incd   [rdi + rax * 4 - 4]
```

```
continue_loop:
```

```
    loop   f_loop
```

```
    pop    rax
```

```
    ret
```

### 3. Файл Makefile

```
all: main

main: main.o lib.o
    gcc main.o lib.o -o main

main.o: main.c
    gcc -c main.c

lib.o: lib.s
    as lib.s -msyntax=intel -mnaked-reg -mmnemonic=intel -o lib.o

clean:
    rm -f *.o main
```