# 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 (пpи его отсутствии получить у пpеподавателя). 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 регистр rax, процедура завершается; если такой интервал не был найден, в rax 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 rcx не будет равен нулю (в rcx изначально хранится количество элементов в исходном массиве сгенерированных чисел).

Исходный код программы представлен в приложении А.

На рисунке 1 представлен результат работы программы.

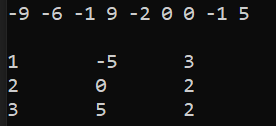


Рисунок1.

## **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;

};

1. 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

1. 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