ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»

Кафедра МО ЭВМ

ОТЧЁТ

по лабораторной работе № 6

по дисциплине «Организация ЭВМ и систем»

Тема: Организация связи Ассемблера с ЯВУ на примере программы построения псевдослучайных целых чисел в заданные интервалы.

Студент гр. 1303 _____ Чернуха В.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Разработать программу, генерирующую набор псевдослучайных чисел и подсчитывающую сколько чисел есть в каждом из заданных диапазонов.

Задание.

Вариант 1.

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.

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

принадлежать итервалу [X_{min}, X_{max}]).

Для бригад с нечетным номером: 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 головную программу, написанную на ЯВУ.

Выполнение работы.

Была написана программа , состоящая из 2 модулей, один написан на ЯВУ C++, в нём происходит считывание из консоли, генерация чисел и вывод результата, другой на ассемблере, в нём подсчитывается в каком промежутке сколько находится сгенерированных чисел.

При запуске программы необходимо ввести начальные данные — количество чисел, диапазон, количество интервалов и их значения. Далее с помощью функции rand() генерируются случайные числа в заданном диапазоне. После этого вызывается функция numcount(), в которую передаются массив чисел и интервалов, их размер, массив для записи результата. Функция реализована на языке ассемблера. В функции мы проверяем числа от конца до начала на вхождение в интервал, если оно в него входит то мы увеличиваем счётчик в массиве результатов. Все числа последовательно сравниваются с текущей левой границей интервала, если оно равно ей или больше, то проверяется меньше ли это число следующей границы интервала (если её нет то счётчик увеличивается), если меньше то переходим на метку count для увеличения счётчика чисел, в ином случае мы переходим на метку check2 и проверяем следующую границу. Если мы проверили все границы или увеличили счётчик, то переходим на check — проверяем следующее случайное число.

Выводы

В результате выполнения лабораторной работы была написана программа, состоящая из 2 модулей на языке C++ и на языке ассемблера, которая генерирует числа и считает сколько чисел в каком интервале находятся.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММ

Название файла: main.cpp

```
#include <ctime>
#include <iostream>
#include <random>
#include <fstream>
extern "C" {void numcount(int* arr, int  NumsRat, int* arrLGrInt, int
NInt, int* res); }
int main()
{
    srand(time(NULL));
    int NumRatDat;
    int Xmin;
    int Xmax;
    int NInt;
    std::cout << "Enter NumRatDat:\n";
    std::cin >> NumRatDat;
    while (NumRatDat >= 16 * 1024) {
        std::cout << "Incorrect value, try again:\n";
        std::cin >> NumRatDat;
    }
    std::cout << "Enter Xmin:\n";
    std::cin >> Xmin;
    std::cout << "Enter Xmax:\n";
    std::cin >> Xmax;
    std::cout << "Enter NInt:\n";
    std::cin >> NInt;
    while (NInt >= 24) {
        std::cout << "Incorrect value, try again:\n";
        std::cin >> NInt;
    }
    int* borders = new int[NInt];
    for (int i = 0; i < NInt; i++) {
        std::cout << "Enter left border#" << i + 1 << ":\n";
        std::cin >> borders[i];
        while ((borders[i] < Xmin || borders[i] > Xmax) || (i > 0 &&
borders[i] <= borders[i - 1])) {
            std::cout << "Incorrect value, try again:\n";
            std::cin >> borders[i];
        }
    }
    int* gennumbers = new int[NumRatDat];
    for (int i = 0; i < NumRatDat; i++) {
        gennumbers[i] = Xmin + rand() % (Xmax - Xmin + 1);
    }
    for (int i = 0; i < NumRatDat; i++) {
        std::cout << gennumbers[i] << " ";
    }
    std::cout << "\n";
    int* res = new int[NInt] {0};
    numcount(gennumbers, NumRatDat, borders, NInt, res);
    std::ofstream fl("res.txt");
    for (int i = 0; i < NInt; i++) {
```

```
        std::cout <<"left border "<< borders[i] << " have " << res[i]
<<" numbers" << "\n";
        fl << "left border#"<<i+1 << ": " << borders[i] << " -- have "
<< res[i] << " numbers" << "\n";
    }
}
```

Название файла: AsmSource.asm

```
.586p
.MODEL FLAT, C
.CODE
numcount PROC C USES EDI ESI, arr:dword, NumsRat:dword, arrLGrInt:dword,
NInt:dword, res:dword

push eax
push ebx
push ecx
push edi
push esi

mov esi, arr
mov ecx, NumsRat
mov edi, arrLGrInt
mov eax, 0

      mov ecx, NumsRat
      check:
            dec ecx
            mov eax, 0
            cmp ecx, -1
            je endprog
            mov ebx, [esi + 4*ecx]; new num
         check2:
            cmp eax, NInt
            je check

            cmp ebx, [edi + 4*eax] ; new left border
         jge checkjg

            inc eax
            jmp check2

            checkjg:
         inc eax
            cmp eax, NInt;-1 ;if -1
            je count

            cmp ebx, [edi + 4*eax];
            jl count
      jmp check2

            count:
            push ebx
            push edi
            mov edi, res
            dec eax
            mov ebx, [edi + 4 * eax]
```

```
            inc ebx
            mov [edi + 4 * eax], ebx
            pop edi
            pop ebx
            jmp check

            endprog:
pop esi
pop edi
pop ecx
pop ebx
pop eax

ret

numcount ENDP
END
```