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

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Организация ЭВМ и систем»

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

Студентка гр. 9383 _____ Сергиенкова А.А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Освоить основные принципы работы организацией связи Ассемблера с ЯВУ. Реализовать программу, которая строит частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Текст задания.

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.

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 целыхчисел - NumRanDat (<= 16K, K=1024)

2. Диапазонизмененияmассивапсевдослучайныхцелыхчисел[Xmin, Xmax] , значения могут быть биполярные; 14

3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - NInt (<=24)

4. Массив левых границ интервалов разбиения LGrInt (должны принадлежать интервалу [Xmin, Xmax]).

Результаты:

2

Текстовый файл, строка которого содержит: - номер интервала, - левую границу интервала, - количество псевдослучайных чисел, попавших в интервал. Количество строк равно числу интервалов разбиения.

Ход работы.

Для реализации ввода начальных данных, а также пользовательского интерфейса, в файле main.cpp пользователь вводит длину массива с псевдослучайными числами, минимум и максимум значений элементов массива, количество интервалов, а также нижние границы интервалов.

Затем в программе массив заполняется псевдослучайными числами, которые равномерно распределяются.

Также создаётся массив, в который будет записано число попаданий чисел в интервал.

Вызывается ассемблерный модуль, в котором происходит распределение количества попаданий псевдослучайных целых чисел в заданные интервалы. Происходит цикличный проход, в процессе которого делается сравнение элементов с левыми границами.

Вывод результата происходит в файле main.cpp в поток и в файл.

Тест.

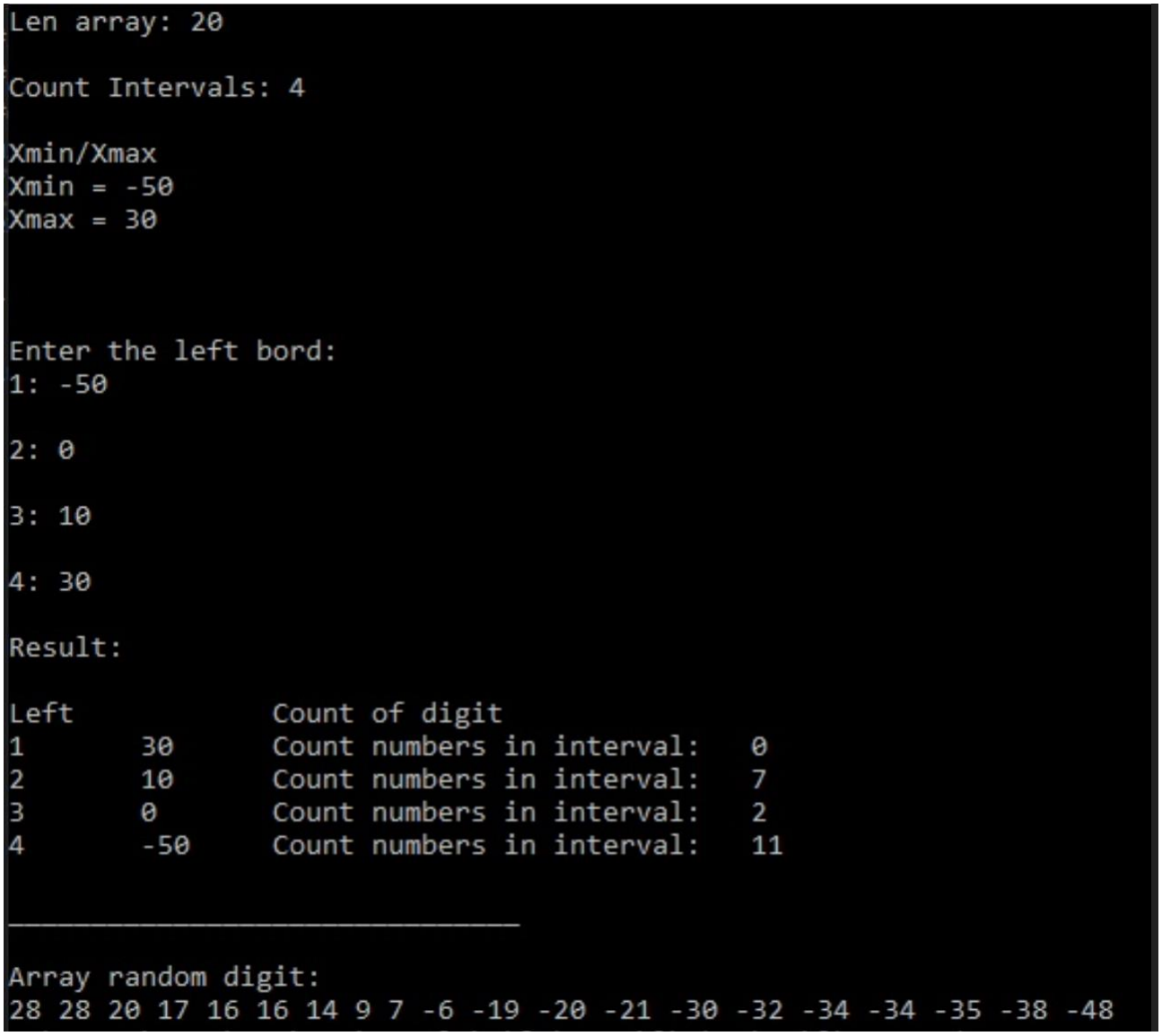

Рисунок 1 - Тест программы

Исходный код программы представлен в приложении А.

Выводы.

Были освоены принципы работы организации связи Ассемблера с ЯВУ. Также была реализована программа, которая позволяет строить частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

main.cpp

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <time.h>
#include <random>
#define PATH "..."
using namespace std;
// 1
extern "C" {
    void AFunc(short* res, short* LGrInt, short* arr, short NInt, short Num-
RanDat);
}

int main() {
    srand(time(0));

    short* arr;                          // массив заданных чисел
    short NumRanDat = 0;                 // длина массива

    short Xmin = 0;                      // диапазон
    short Xmax = 0;

    short* LGrInt;                       // массив левых границ
    short NInt = 0;                      // кол-во интеравалов
    short* res;                          // кол-во чисел в каждом интеравле

    ofstream out(PATH);                  // файл

    cout << "Len array: ";
    cin >> NumRanDat;
    while (NumRanDat > 16000 || NumRanDat < 0){
        cout << "Invalid len" << endl;;
        cin >> NumRanDat;
    }
    cout << endl;

    arr = new short[NumRanDat];
    //----------------------------------------
    cout << "Count Intervals: ";
    cin >> NInt;
    while(NInt > 24 || NInt < 1){
        cout << "Invalid count" << endl;;
        cin >> NInt;
    }
    cout << endl;

    LGrInt = new short[NInt];
    //----------------------------------------
    cout << "Xmin/Xmax" << endl;
    cout << "Xmin = ";
    cin >> Xmin;
```

```
cout << "Xmax = ";
cin >> Xmax;

cout << endl;

while(Xmax <= Xmin) {
    cout << "Invalid Xmin/Xmax" << endl;;
    cout << "Xmin = ";
    cin >> Xmin;

    cout << "Xmax = ";
    cin >> Xmax;
}

cout << endl;
//-----------------------------------------

// заполним остальные границы
cout << "\nEnter the left bord: " << endl;
cout << "1: " << Xmin << endl << endl;
LGrInt[0] = Xmin;

for(int i = 1; i < NInt; ++i){
    do{
        cout << i + 1 << ": ";
        cin >> LGrInt[i];
        cout << endl;
    } while(LGrInt[i] < Xmin || LGrInt[i] > Xmax);
}

// отсортировали массив границ
for(int i = 0; i < NInt - 1; ++i){
    for (int j = 0; j < NInt - i - 1; ++j){
        if(LGrInt[j] < LGrInt[j + 1]) {
            short t = LGrInt[j];
            LGrInt[j] = LGrInt[j + 1];
            LGrInt[j + 1] = t;
        }
    }
}
//-----------------------------------------

// равномерно распределим
for(int i = 0; i < NumRanDat; ++i){
    for(int i = 0; i < NumRanDat; ++i)
        arr[i] = Xmin + rand() % (Xmax - Xmin);
}
// sort
for(int i = 0; i < NumRanDat - 1; i++) {
    for(int j = 0; j < NumRanDat - i - 1; j++) {
        if (arr[j] < arr[j + 1]) {
            short t = arr[j];
            arr[j] = arr[j + 1];
            arr[j + 1] = t;
        }
    }
}
```

```
    res = new short[NInt];
    for(int i = 0; i < NInt; ++i){
        res[i] = 0;
    }

    AFunc(res, LGrInt, arr, NInt, NumRanDat);

    out << "Result:\n" << endl;
    out << "Left\t\tCount of digit" << endl;
    cout << "Result:\n" << endl;
    cout << "Left\t\tCount of digit" << endl;
    for(int i = 0; i < NInt; ++i){
        out << i + 1 << '\t' << LGrInt[i] << '\t' << "Count numbers in
interval:  " << res[i] << endl;
        cout << i + 1 << '\t' << LGrInt[i] << '\t' << "Count numbers in
interval:   " << res[i] << endl;
    }
    cout << endl;
    cout << "_____________________________" << endl;
    out << endl;
    out << "_____________________________" << endl;

    // сортируем рандомные числа
    for(int i = 0; i < NumRanDat - 1; ++i){
        for(int j = 0; j < NumRanDat - i - 1; ++j){
            if(arr[j] < arr[j + 1]){
                short t = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = t;
            }
        }
    }
    cout << endl;
    out << endl;
    cout << "Array random digit: " << endl;
    out << "Array random digit: " << endl;
    for(int i = 0; i < NumRanDat; ++i){
        cout << arr[i] << ' ';
        out << arr[i] << ' ';
    }


    out.close();
    delete[] LGrInt;
    delete[] res;
    delete[] arr;

    return 0;
}
```

Файл lb6.asm

```
.MODEL FLAT, C
 .DATA
 item DW 0
 .CODE

AFunc PROC C

       mov  eax,[esp+4]                    ; res                  (к смещению
прибавляем одно и тоже число, тк типы данных одинаковые)
       mov  ebx,[esp+8]                    ; LGrInt
       mov  edx,[esp+12]                   ; arr
       mov  edi, [esp+16]                  ; NInt
       mov  cx, [esp+20]                   ; NumRanDat

       and ecx,0000ffffh                   ; обнуляем верхние 16 бит
(счетчики)
       and edi,0000ffffh



FirstFunc:

       push ax                                    ; заталкиваем ax в стек
       mov ax, [edx]                       ; в ax текущий элемент массива
из псевдослучайных чисел
       mov item, ax
       pop ax

       mov esi, 0                                 ; esi - индекс массива
res
       push ecx
       mov ecx, edi                        ; счетчик



SecondFunc:

       push ebx

       mov bx, [ebx+esi]                   ; положили в bx элемент из
массива LGrInt (взяли смещение ebx(LGrInt массив) + индекс массива res)
       cmp item, bx                        ; сравнили элемент с элементом
из массива псевдослучайных чисел

       jge ThirdFunc                       ; если больше или равно, то
подходит для интервала
       add esi, 2                                 ; увеличиваем индекс на 2

       pop ebx

       loop SecondFunc                            ; цикл закончится когда
пройдем все интервалы
```

```
ThirdFunc:

        mov bx, [eax+esi]                    ; положили в bx элемент из
массива количества элементов (взяли смещение eax(res массив) + индекс res)
        inc bx
        mov [eax+esi],bx                     ; увеличиваем кол-во, т.к нашли
элемент в границе

        pop ebx                                      ; вытаскиваем ebx(LGrInt)
        pop ecx                                      ; счетчик для FirstFunc

        add edx,2                                    ; сдвигаемся на 2

        loop FirstFunc                       ; внешний цикл


ret

AFunc ENDP
END
```