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 распределения попаданий псевдослучайных целых чисел в заданные интервалы

Студентка гр. 1381 _____ Новак П.И.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Научиться связывать язык Ассемблера и язык высокого уровня так, чтобы функции ассемблерного модуля вызывались из программы на ЯВУ. Написать программу построения частного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Задание.

На языке высокого уровня программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение.

Далее должны вызываться две 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.

Исходные данные:

1. Длина массива псевдослучайных целых чисел – *NumRanDat*
2. Диапазон изменения массива псевдослучайных целых чисел $[XX_{mmiiii}, XX_{mmmmm}]$, могут быть биполярные;
3. Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел - *NInt* (<=24)
4. Массив левых границ интервалов разбиения *LGrInt* (должны принадлежать интервалу $[XX_{mmiiii}, XX_{mmmmm}]$).

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.

Программа написана с использованием языка программирования C++. В файле *lab6.cpp* написаны функции для сортировки массива *sort*, для вывода информации о массиве *print, print_task,* а так же *main*, где происходит считывание значений для задания интервалов, а так же их размерности. Вызывается функция, написанная на языке Ассемблер, в ней изначально происходит сохранение значений используемых регистров в стек, затем в блоке *ans* происходит сортировка чисел по интервалам. Пока числа массива подходят под критерии интервала, программа находится в блоке *count*, если число не подходит под критерии - совершается прыжок в блок *out.* Программа остаётся в *ans,* пока не закончится *loop,* то есть пока регистр cx>0. Затем происходит восстановление регистров из стека.

Результат работы программы выводится в файл out.txt.

Исходный код программы представлен в приложении А.

Рис. 1 - Тест 1: числа на отрезке [1,100]

Результат тестирования представлен на рисунке 1.

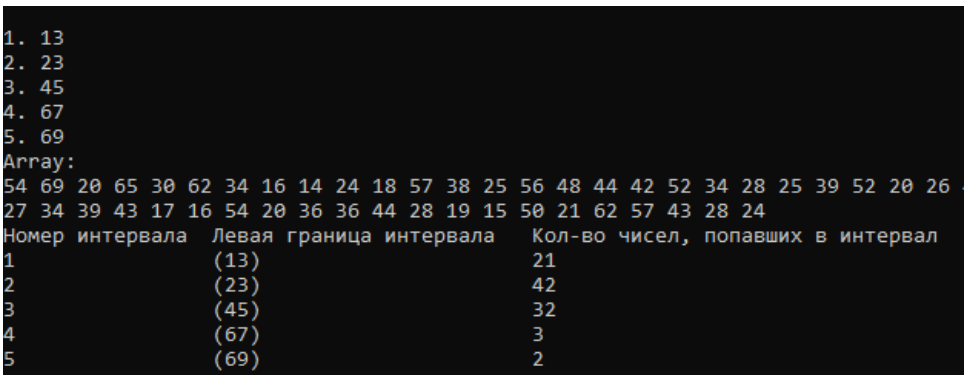

Рис. 1 - Тест 1: числа на отрезке [1,100]

Выводы.

Изучены принципы организации связи Ассемблера с ЯВУ, а также разработана программа, которая строит частотное распределение попаданий псевдослучайных целых чисел в заданные интервалы.

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

Название файла: lab6.cpp

```
//#include"stdafx.h"
#include"iostream"
//#include<iomanip>
#include <fstream>

using namespace std;
std::ofstream file1("out.txt");

extern "C" {void function(int* array, int len, int* LGrInt, int NInt,
int* answer); }

void sort(int*arr, int count_) {
for (int i = 0; i < count_ - 1; i++) {
for (int j = 0; j < count_ - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
int b = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = b;
}
}
}
}

void print(int*arr, int count_) {
cout << endl << endl;
for (int j = 0; j < count_; j++) {
cout << j + 1 << ". " << arr[j] << "\n";
}
}

void print_task(int*& arr, int*& LGr, int count_, int length_, int*&
result_final) {
std::cout << "Array:\n";
for (int i = 0; i < length_; i++)
{
std::cout << arr[i] << " ";
}
std::cout << "\n";
file1 << "\n";
cout << "Номер интервала\t Левая граница интервала   Кол-во чисел,
попавших в интервал\n";
file1 << "Номер интервала\t Левая граница интервала   Кол-во чисел,
попавших в интервал\n";
for (int j = 0; j < count_; j++) {
cout << j + 1 << "\t\t (" << LGr[j] << ")\t\t\t   " << result_final[j] <<
"\n";
file1 << j + 1 << "\t\t (" << LGr[j] << ")\t\t\t   " << result_final[j]
<< "\n";
}
}
```

```
int main() {
setlocale(0, "RUS");

int length_; //длина массива псевдослучайных чисел

do {
cout << "Введите длину массива псевдослучайных чисел\n";
cin >> length_;
} while ((length_ > 16000) || (length_ <0));

int Xmin, Xmax;  //левая и правая границы диапазона изменения массива

do {
cout << "Введите минимальное значение отрезка\n";
cin >> Xmin;
cout << "Введите максимальное значение отрезка\n";
cin >> Xmax;
} while (Xmin > Xmax);

int*arr = new int[length_];  //массив псевдослучайных чисел

for (int i = 0; i < length_; i++) {
arr[i] = Xmin + rand() % (Xmax - Xmin + 1);
};

file1 << "Массив псевдо-случайных чисел:\n";
for (int i = 0; i < length_; i++)
{
file1 << arr[i] << " ";
}
file1 << "\n";

int count_;  //количество интервалов разбиения
do {
cout << "Введите количество интервалов разбиения\n";
cin >> count_;
} while ((count_ >24) || (count_ < 1) || (count_ >(Xmax - Xmin + 1)));

int*LGrInt = new int[count_];  //массив левых границ
int i = 0;

cout << "Введите левые границы интервалов разбиения\n"" 1-я граница -
начало отрезка (" << Xmin << ") \n\n";
LGrInt[i] = Xmin;
for (int j = 1; j < count_; j++) {
do {
cout << "Введите " << j + 1 << "-ю границу\n";
cin >> LGrInt[j];
} while ((LGrInt[j] < Xmin) || (LGrInt[j] > Xmax));
}
sort(LGrInt, count_);

int* result_final = new int[count_] {0};//массив результатов заполняем 0

function(arr, length_, LGrInt, count_, result_final);

print(LGrInt, count_);
```

```
print_task(arr, LGrInt, count_, length_, result_final);
system("pause");

delete[] arr;
delete[] LGrInt;
delete[] result_final;
file1.close();
return 0;
}
```

Название файла: lab6_2.asm

```
.586p
.MODEL FLAT, C
.CODE
function PROC C USES EDI ESI, array:dword, len:dword, LGrInt:dword,
NInt:dword, answer:dword

push eax
push ebx
push ecx
push edi
push esi

mov ecx, len
mov esi, array
mov edi, LGrInt
mov eax, 0

ans:
mov ebx, 0 ;текущее количество пройденных интервалов
count:
 cmp ebx, NInt ;сравнение ebx и кол-ва интервалов разбиения
 jge out ;сли >= выходим в другой блок

 push eax
 mov eax, [esi + 4 * eax] ;в eax кладутся поочерёдно элементы массива,
масшт на 4 в силу размера
 cmp eax, [edi + 4 * ebx] ;сравниваются этот эл-т массива и эл-т в
зависимости от кол-ва пройденных интервалов
 pop eax ;возвращается eax
 jl out ;если < то выходим
 inc ebx ;иначе меняется интервал
 jmp count ;снова идем в этот блок

out:
 dec ebx ;возврат к предыдущему интервалу

 cmp ebx, -1
 je to_next_num ;если отрицательный идем в блок next_sum
 mov edi, answer
 push eax
 mov eax, [edi + 4 * ebx] ;помещается ebx-й по счёту эл-т в массиве
результатов
 inc eax ;+1
 mov [edi + 4 * ebx], eax ;в ebx-ю ячейку массива рез-в помещается eax
```

```
 pop eax
 mov edi, LGrInt

to_next_num:
inc eax

loop ans

pop esi
pop edi
pop ecx
pop ebx
pop eax

ret

function ENDP
END
```