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.

Вариант 22

Студент гр. 1383 _____ Харитонов Н.М

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Научиться работать с ассемблерными модулями, создавать их и запускать их языка высокого уровня на примере языка c++;

Задание.

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.

Вариант 22.

Для бригад с четным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде двух ассемблерных модулей, первый из которых формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу на ЯВУ как промежуточный результат. Это распределение должно выводиться в текстовом виде для контроля. Затем вызывается второй ассемблерный модуль, который по этому промежуточному распределению формирует окончательное распределение псевдослучайных целых чисел по интервалам произвольной длины (с заданными границами). Это распределение возвращается в головную программу и выдается как основной результат в виде текстового файла и, возможно, графика.

Выполнение работы.

Создан файл lr6.asm, в ней содержится функция func1, которая принимает массив с исходными псевдослучайными числами, его длину, массив, в который будут записываться повторения всех возможных чисел и минимальное число диапазона. Эта подпрограмма подсчитывает и записывает в массив по индексу количество одинаковых чисел.

В файле lr6_2.asm находится функция func2, которая принимает массив, который получился в 1-й подпрограмме, массив с левыми границами, массив, в который планируется записать количество вхождений чисел в диапазон, размер первого массива, размер второго и третьего массива (их размер одинаковый) и минимальное число диапазона.

Программа принимает на вход размер массива псевдослучайных чисел, левую границу массива, правую границу массива, количество разбиений и левые границы разбиений. После этого идет генерация чисел и выполнение func1. Затем вызывается функция func2 и, после нее, идет вывод результата работы программы в консоль, а именно, левая граница разбиения и количество вхождений чисел в диапазон от нее до следующей левой границы разбиения или до правой границы чисел диапазона.

Выводы.

В ходе выполнения работы были изучены ассемблерные модули и разработана программа с их использованием на языке высокого уровня (c++).

ПРИЛОЖЕНИЕ А
КОД ПРОГРАММЫ

main.cpp:

```
#include <iostream>
#include <windows.h>
#include <fstream>
#include <random>

using namespace std;

extern "C" void(func1(int* array, int arr_size, int*
raspr_arr, int Xmin));
extern "C" void(func2(int* arr1, int* arr2, int* res,
int arr1_len, int index,int Xmin));

int comparator(const void* v1, const void* v2) {
    int i1 = *((int*)v1);
    int i2 = *((int*)v2);
    return i1 - i2;
}

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    int size;
    int count;
    int Xmin;
    int Xmax;
    cout << "Введите размер массива" << '\n';
```

```
cin >> size;
int* arr = new int[size];

cout << "Введите левую границу массива" << '\n';

cin >> Xmin;

cout << "Введите правая граница массива" << '\n';

cin >> Xmax;

if (Xmin> Xmax) {
    cout << "Неверные границы массива";
    return 1;
}


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

cout << "Введите количество разбиений" << '\n';
cin >> count;

int *LGInt = new int[count];
cout << "Введите разбиения" << '\n';
```

```
for (int i = 0; i < count; i++)
{
    cin >> LGInt[i];
}
qsort(LGInt, count, sizeof(int), comparator);
// конец ввода
//-----------------------

cout << "Сгенерированные числа:" << '\n';
for (int i = 0; i < size; i++)
{
    cout << arr[i] << ' ';
}
int raspr_len = Xmax - Xmin + 1;
cout << '\n' << "Массив распределения" << '\n';

int* mas_raspr = new int[raspr_len];
for (int i = 0; i < raspr_len; i++)
{
    mas_raspr[i] = 0;
}
func1(arr, size, mas_raspr, Xmin);
for (int i = 0; i < raspr_len; i++)
{
    cout << mas_raspr[i] << ' ';
}
int* res_mas = new int[count];
for (int i = 0; i < count; i++)
{
    res_mas[i] = 0;
```

```
        }
        func2(mas_raspr,   LGInt,   res_mas,   raspr_len,
count, Xmin);
        cout << "\n\n";
        for (int i = 0; i < count; i++)
        {
            cout << LGInt[i] << ": " << res_mas[i] <<
'\n';
        }
        /*
        for (int i = 0; i < size; i++)
        {
            std::cout << arr[i] << ' ';
        }
        std::cout << '\n';
        func(arr, size);
        for (int i = 0; i < size; i++)
        {
            std::cout << arr[i] << ' ';
        }
        */
    }
```

lr6.asm:

```
.model flat
.code
public C func1
func1 proc PROC C array:dword, arr_size:dword, raspr_arr:dword, Xmin:dword

push eax
push ebx
push ecx
push edx
push edi
push esi

mov esi, array
mov edi, raspr_arr
mov ecx, 0
mov eax, 0
mov ebx, 0
loop1:
    mov eax, [esi][ecx*4]
    sub eax, Xmin
    mov ebx, [edi+4*eax]
    add ebx, 1
    mov [edi+4*eax], ebx
    ;iteration
    add ecx, 1
    cmp ecx, arr_size
    jl loop1
```

```
pop esi
pop edi
pop edx
pop ecx
pop ebx
pop eax

ret

func1 ENDP
END
```

lr6_2.asm:

```
.model flat
.code

public C func2
func2 PROC C arr1:dword, arr2:dword, res:dword,
arr1_len:dword, index:dword, Xmin:dword

push eax
push ebx
push ecx
push edx
push esi
push edi

mov esi, arr1
mov edi, arr2
mov ecx, arr1_len ;i
mov edx, index

sub ecx, 1
sub edx, 1

loop1:
    cmp edx, 0
    jl fin
    mov ebx, [edi][edx*4]
    sub ebx, Xmin
    cmp ecx, ebx
    jl met1
```

```
        push edi
        mov edi, res
        mov ebx, [esi][ecx*4]
        mov eax, [edi+edx*4]
        add eax, ebx

        mov [edi+edx*4], eax
        pop edi
        jmp iter
    met1:
        sub edx, 1
        add ecx, 1
    iter:
    ;iteration
    sub ecx, 1
    cmp ecx, 0
    jge loop1

fin:


pop edi
pop esi
pop edx
pop ecx
pop ebx
pop eax
ret
func2 endp
End
```