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.

Студент(ка) гр. 1303 _____ Хулап О. А.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2022

Цель работы

Написать программу, связывающую Ассемблер и ЯВУ.

Общая формулировка задачи

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 случаеинтервалы 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. Диапазон изменения массива псевдослучайных целых чисел [Xmin, Xmax], значения 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 [Xmin, Xmax]).

Выполнение работы

Вариант 2

В ходе выполнения лабораторной работы написана программа на языке C++, производящая считывание входных данных и запись результата в файл и в консоль.

Написано два модуля на языке Ассемблера.

Первый модуль (first.asm) формирует распределение исходных чисел по интервалам единичной длины и возвращает его в вызывающую программу как промежуточный результат: по метке l1 производится анализ каждого числа исходного массива (arr) и инкрементируется соответствующее значение промежуточного массива (answer_arr). Индекс соответствующей ячейки промежуточного массива находится путем вычитания из анализируемого числа Xmin.

Второй модуль (second.asm) по метке l1 находит левый и правый индекс анализируемого интервала, и затем по метке l2 складывает значения из промежуточного массива до тех пор, пока не достигнута следующая граница, и записывает в выходной массив (arr_out), затем происходит возвращение к метке l1 и анализируется следующий интервал.

Оба модуля подключены в программу из файла lb6.cpp.

Исходный код в приложении А.

Выводы

В ходе выполнения лабораторной работы были приобретены знания о написании программ, связывающий Ассемблер и ЯВУ, а также была написана программа, реализующая эту связь.

ПРИЛОЖЕНИЕ А

Файл lb6.cpp

```
#include <iostream>
#include <fstream>
#include <random>

std::ofstream file;

extern "C" {void first(int* array, int len, int x_min, int
x_max, int* arr_out); }
extern "C" {void second(int* array, int NInt, int x_min, int*
gr, int* arr_out); }

void generate_array(int*& arr, int NumRamDat, int min, int max);

void receive(int& NumRamDat, int*& arr, int& min, int& max, int&
NInt, int*& LGrInt) {
    std::cout << "Enter the length of the array:" << std::endl;
    std::cin >> NumRamDat;
    arr = new int[NumRamDat];
    std::cout << "Enter limits of random numbers:" << std::endl
<< "Min:";
    std::cin >> min;
    std::cout << "Max:";
    std::cin >> max;

    while (min >= max) {
        std::cout << "Invalid Xmax, enter Xmax again:";
        std::cin >> max;
    }

    generate_array(arr, NumRamDat, min, max);

    std::cout << "Enter number of intervals:";
    std::cin >> NInt;

    while (NInt < 0 || NInt > 24) {
        std::cout << "Invalid NInt, please enter NInt again:";
        std::cin >> NInt;
    }

    LGrInt = new int[NInt + 1];
    std::cout << "Enter intervals:" << std::endl;

    for (int i = 0; i < NInt; i++)
    {
        std::cout << "Interval " << i + 1 << ": ";
        std::cin >> LGrInt[i];

        while (LGrInt[i] > max || LGrInt[i] < min) {
```

```
            std::cout << "Invalid interval, please enter the
interval again:";
            std::cin >> LGrInt[i];
        }
    }
    LGrInt[NInt] = max + 1;
    std::sort(LGrInt, LGrInt + NInt);
}

void generate_array(int*& arr, int NumRamDat, int min, int max)
{
    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dist(min, max);

    for (int i = 0; i < NumRamDat; i++) {
        arr[i] = dist(gen);
    }
}

void print(int NInt, int NumRamDat, int*& arr, int*& LGrInt,
int*& answer) {
    std::cout << "Generated array:" << std::endl;

    for (int i = 0; i < NumRamDat; i++)
    {
        std::cout << arr[i] << " ";
    }

    std::cout << std::endl;
    file << std::endl;
    std::cout << "Index\t" << "Border\t" << "Count" <<
std::endl;
    file << "Index\t" << "Border\t" << "Count" << std::endl;

    for (int i = 0; i < NInt; i++) {
        std::cout << "  " << i + 1 << "\t  " << LGrInt[i] << "\t
" << answer[i] << std::endl;
        file << "  " << i + 1 << "\t  " << LGrInt[i] << "\t  "
<< answer[i] << std::endl;
    }
}

int main(){
    file.open("answer.txt", std::ios_base::out);

    int NumRamDat;
    int Xmin;
    int Xmax;
    int NInt;
    int* arr;
    int* LGrInt;
```

```
    receive(NumRamDat, arr, Xmin, Xmax, NInt, LGrInt);

    int* answer_arr = new int[Xmax - Xmin + 1]{ 0 };
    int* arr_out = new int[NInt] {0};

    first(arr, NumRamDat, Xmin, Xmax, answer_arr);
    std::cout << std::endl;


    second(answer_arr, NInt, Xmin, LGrInt, arr_out);
    std::cout << std::endl;

    print(NInt, NumRamDat, arr, LGrInt, arr_out);

    delete[] arr;
    delete[] LGrInt;
    delete[] answer_arr;

    file.close();
}
```

Файл first.asm

```
.586p
.MODEL FLAT, C
.CODE
first PROC C USES EDI ESI, arr:dword, len:dword, x_min:dword,
x_max:dword, arr_out:dword

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

mov esi, arr ; esi - input array address
mov ecx, len ; ecx - input array size
mov edi, arr_out ; edi - output array address
mov eax, x_max
sub eax, x_min
inc eax

mov ebx, eax
mov edx, 0
mov eax, 0


mov edx, 0
l1:
     mov eax, [esi + 4 * edx]
     sub eax, x_min
```

```
     mov ebx, [edi + 4 * eax]
     inc ebx
     mov [edi + 4 * eax], ebx
     inc edx
     dec ecx
     cmp ecx, 0
     jne l1

pop edi
pop esi
pop edx
pop ecx
pop ebx
pop eax
ret

first ENDP
END
```

Файл second.asm

```
.586p
.MODEL FLAT, C
.CODE
second PROC C USES EDI ESI, array:dword, NInt:dword,
x_min:dword, gr:dword, arr_out:dword

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

mov esi, array ; esi - input array address
mov ecx, NInt ; ecx - border array size
mov edi, gr ; edi - border array
mov ebx, 0

l1:
     mov eax, [edi + 4 * ebx] ; eax - left border
     sub eax, x_min ; eax - left index
     inc ebx
     mov edx, [edi + 4 * ebx] ; edx - right border
     sub edx, x_min ; edx - right index
     push ecx
     mov ecx, 0

     l2:
          add ecx, [esi + 4 * eax]
          inc eax
          cmp eax, edx
          jl l2
```

```
        dec ebx
        push edi
        mov edi, arr_out
        mov [edi + 4 * ebx], ecx
        inc ebx
        pop edi
        pop ecx
        dec ecx
        jnz l1


pop edi
pop esi
pop edx
pop ecx
pop ebx
pop eax
ret

second ENDP
END
```