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 14

Студентка гр. 1383 _____ Манучарова А.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2022

Цель работы.

Научиться писать программы на ЯВУ с использованием ассемблерных модулей.

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 14.

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.

//f1, f2

Была создана функция func1 в файле f1.asm, которая принимает 4 параметра: массив со случайными числами, размер этого массива, массив с распределением чисел, минимальное число из диапазона. Она делает так, что каждое вхождение определенного числа из исходного массива прибавляет соответствующее значение по индексу у второго массива.

Была создана функция func2 в файле f2.asm, которая принимает 6 параметров: массив с распределением чисел, массив с левыми границами, массив с подсчетом всех вхождений чисел в диапазон, длина массива с распределением, длина двух последующих массивов и минимальное значение диапазона.

Программа принимает на вход количество псевдослучайных чисел NumRanDat, левую границу диапазона Xmin, правую границу Xmax, количество левых границ NInt и значения левых границ LGrInt. Печатает некоторые промежуточные результаты: сгенерированные числа, массив распределения. Также выводит результирующий массив.

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* arr, int NumRanDat, int* res, int Xmin));
extern "C" void(func2(int *res, int*LGrInt, int*result, int interval, int NInt, int Xmin));

int cmp(const void* num1, const void* num2) {
    int* first = (int*)num1;
    int* second = (int*)num2;
    if (*first > *second)
        return 1;
    else if (*first < *second)
        return -1;
    else
        return 0;
}

void print_arr(int* arr, int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
```

```
        printf("\n");
    }


    int main() {
        int NumRanDat, Xmin, Xmax, NInt;
        printf("Enter NumRanDat, Xmin, Xmax, NInt:");
        scanf_s("%d %d %d %d", &NumRanDat, &Xmin, &Xmax,
&NInt);

        if (NumRanDat < 0 || NumRanDat>16384) {
            printf("Wrong NumRanDat");
            return 0;
        }
        if (Xmin > Xmax) {
            printf("Wrong Xmin and Xmax");
            return 0;
        }
        if (NInt > 24 || NInt < 0) {
            printf("Wrong NInt");
            return 0;
        }

        printf("Enter left borders: ");
        int* LGrInt = new int[NInt];
        for (int i = 0; i < NInt; i++)
        {
            cin >> LGrInt[i];
        }
        qsort(LGrInt, NInt, sizeof(int), cmp);
```

```
        if (LGrInt[0] < Xmin || (LGrInt[NInt - 1] > Xmax
&& NInt > 0)) {
            printf("Wrong border");
            delete[] LGrInt;
            return 0;
        }

        int* arr = new int[NumRanDat];
        srand(time(NULL));
        for (int i = 0; i < NumRanDat; i++)
            arr[i] = Xmin + rand() % (Xmax - Xmin + 1);

        printf("Random numbers array: \n");
        print_arr(arr, NumRanDat);
        int interval = Xmax - Xmin + 1;
        int* res = new int[interval];
        for (int i = 0; i < interval; i++)
        {
            res[i] = 0;
        }
        func1(arr, NumRanDat, res, Xmin);
        print_arr(res, interval);

        int* result = new int[NInt];
        for (int i = 0; i < NInt; i++)
        {
            result[i] = 0;
        }
        func2(res, LGrInt, result, interval, NInt, Xmin);
        printf("result: \n");
```

```
        ofstream out;
        out.open("res.txt");
        for (int i = 0; i < NInt; i++) {
            printf("%d - %d\n", LGrInt[i], result[i]);
            out << LGrInt[i] << " - " << result[i] <<
"\n";
        }
        out.close();
        delete[] arr;
        delete[]LGrInt;
        delete[]res;
        delete[]result;
        return 0;
    }
```

a1.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
```

f2.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
```