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

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 гр. 9383 _____ Хотяков Е.П.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Научиться связывать программу на ЯВУ с ассемблерными модулями. Написать программу на основе изученного.

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. Исходные данные. 1. Длина массива псевдослучайных целыхчисел - NumRanDat (<= 16K, K=1024) 2. Диапазон изменения массива псевдослучайных целых чисел [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: 1. Текстовый файл, строка которого содержит: - номер интервала, - левую границу интервала, - количество псевдослучайных чисел, попавших в интервал. Количество строк равно числу интервалов разбиения. 2. График, отражающий распределение чисел по интервалам. (необязательный результат) В зависимости от номера бригады формирование частотного распределения должно производиться по одному из двух вариантов:

2. 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.

Ход работы:

Для реализации поставленной задача была написана программа, состоящая из трех файлов:

main.cpp — в функции main происходит генерация псевдослучайных целых чисел, их обработка с помощью вызова ассемблерных модулей и вывод результатов в консоль и текстовый файл;

lab6_module1.asm – первый ассемблерный модуль, в котором происходит заполнение массива array_mod1 вхождений каждого сгенерированного числа по интервалам единичной длины;

lab6_module2.asm – второй ассемблерный модуль, в котором происходит заполнение массива array_mod2 на количество вхождений каждого элемента в интервалы, определенные пользователем;

Вывод:

Произошло ознакомление со связыванием программы на ЯВУ с ассемблерными модулями. Написана программа на основе изученного.

Приложение А

Main.cpp:

```
#include <iostream>
#include <cmath>
#include <fstream>
using namespace std;

extern "C" {
void module_1(int* mass, int NumRatDat, int* mass_after_module1, int Xmin);
void module_2(int* mass_after_module1, int NumRatDat, int* mass_after_module2, int Xmin, int*
LGrInt, int Nint);
}


int main(){
unsigned int NumRatDat;//количество чисел для генерации
int Xmin, Xmax;
unsigned int Nint;//количество границ интервала
int *LGrInt;//массив левых границ интервала
int *mass; //массив сгенерированных чисел
int *mass_after_module1; //массив количества каждого сгенерированного числа
int *mass_after_module2; //массив количества сгенерированных чисел в заданных интервалов
ofstream out("./out.txt");
if(!out.is_open()){
cout << "Файл не удалось открыть\n";
exit(1);
}
//Ввод исходных данных:
cin >> NumRatDat;
cin >> Xmin >> Xmax;
cin >> Nint;
LGrInt = new int[Nint];
for(int i = 0; i < Nint; i++){
cin >> LGrInt[i];
}

mass = new int[NumRatDat];
cout << "Сгенерированные числа:\n";
//генерация псевдослучайных чисел
for (int i = 0; i < NumRatDat; i++){
mass[i] = Xmin + rand() % (Xmax - Xmin + 1);
cout << mass[i] << ' ';
}
cout << '\n';

int length_1 = abs(Xmax - Xmin + 1);//длинная массива mass_after_module1

mass_after_module1 = new int[length_1];
```

```
module_1(mass, NumRatDat, mass_after_module1, Xmin);

cout << "Распределение после работы первого модуля: \n";
out << "Распределение после работы первого модуля: \n";
for(int i = 0; i < length_1; i++){
cout << "Инт№" << i+1 << "\tЛевая граница:" << i+1 << "\tКоличество чисел:" <<
mass_after_module1[i] << '\n';
out << "Инт№" << i+1 << "\tЛевая граница:" << i+1 << "\tКоличество чисел:" <<
mass_after_module1[i] << '\n';
}

module_2(mass_after_module1, NumRatDat, mass_after_module2, Xmin, LGrInt, Nint);

cout << "Распределение после работы первого модуля: \n";
out << "Распределение после работы первого модуля: \n";
for(int i = 0; i < Nint; i++){
cout << "Инт№" << i+1 << "\tЛевая граница:" << LGrInt[i] << "\tКоличество чисел:" <<
mass_after_module2[i] << '\n';
out << "Инт№" << i+1 << "\tЛевая граница:" << LGrInt[i] << "\tКоличество чисел:" <<
mass_after_module2[i] << '\n';
}



return 0;
}
```

lab6_module1.asm

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C module_1
module_1 PROC C mass: dword, NumRatDat: dword, mass_after_module1: dword, Xmin: dword

push esi
push edi

mov edi, mass ;исходный массив
mov ecx, NumRatDat ;размер исходного массива
mov esi, mass_after_module1 ;массив на выход

module_1_work:
mov eax, [edi]
sub eax, Xmin
mov ebx, [esi + 4*eax]
inc ebx
mov [esi + 4*eax], ebx
loop module_1_work
```

```
pop edi
pop esi

ret
module_1 ENDP
END
```

lab6_module2.asm

```
.586p
.MODEL FLAT, C
.CODE
PUBLIC C module_2
module_2 PROC C mass_after_module1: dword, NumRatDat: dword, mass_after_module2: dword,
Xmin: dword, LGrInt: dword, Nint: dword

push esi
push edi

mov edi, mass_after_module1 ;исходный массив
mov ecx, Nint ;количество интервалов
mov esi, LGrInt ;массив левых границ

change_interval:
mov eax, [esi]
sub eax, Xmin
mov [esi], eax
add esi, 4
loop change_interval

mov esi, LGrInt ;смещаем на начало массива
mov ecx, Nint ;обновляем счетчик
mov ebx, 0 ;Смещение относительно начала массива

move eax, [esi];сколько шагов будем делать при первом проходе(берем первый левый конец
интервала)
add_1:
push ecx ;сохраняем счетчик
mov ecx, eax ;запускаем новый счетчик
push esi ; сохраняем массив левых границ
mov esi, mass_after_module2 ;записываем в esi итоговый массив
add_2:
mov eax, [edi];записываем текущее значение изначального массива
add [esi + 4*ebx], eax;прибавляем это значение к количеству элементов на текущем интервале
add edi, 4 ;переходим на следующий элемент изначального массива
loop add_2

pop esi
pop ecx
inc ebx
mov eax, [esi];берем текущую левую границу
add esi, 4;берем следующую левую границу
```

```
sub eax, [esi];вычисляем расстояние между границами
neg eax; берем модуль расстояния
loop add_2

;на последнем интервале количество чисел будет равно разности колучества чисел вообще и количества числе, добавленных
mov esi, mass_after_module2
mov ecx, Nint
mov eax, 0

add_3:
add eax, [esi]
add esi, 4
loop add_3

mov esi, mass_after_module2
sub eax, NumRatDat
neg eax

add [esi + 4*ebx], eax

pop edi
pop esi

ret
module_2 ENDP
END
```