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.

Студентка гр. 9383 _____ Орлов Д.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Научиться реализовывать связь Ассемблера и ЯВУ. Написать программу построения частотного распределения попаданий псевдослучайных чисел в заданные интервалы.

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.

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]).

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).

Ход работы.

В ходе работы была реализована программа из 3-х модулей, 1 на C++ (ЯВУ) и 2 других на ассемблере.

На ЯВУ написан main.cpp, который собирает от пользователя входную информацию и перенаправляет ее в ассемблерные модули. Так же здесь осуществляется вывод данных в консоль и файл.

На ассемблере написано 2 модуля. Первый реализует распределение чисел по единичным отрезкам. Это сделано с помощью команды loop. Циклически записывается в новый массив количество повторений каждого числа.

Второй модуль формирует распределение тех же чисел, но уже по заданным интервалам. Это происходит благодаря нескольким циклам, в которых левые границы переводятся в неотрицательные числа и сопоставляются числам с таким же индексом из массива, полученного в первом модуле.

Связь между модулями осуществлена с помощью спецификатора extern, который позволяет выполнять раздельную компиляцию модулей.

Исходный код см. в приложении А.

Тестирование.

№	Исходные данные	Результат
1	NumDatRan=100 xmin=-50 xmax=50 NInt=10 LGrInt={-50, -43, -30, 0, 10, 12, 15, 30, 40, 49 }	№ Лев.гр. Кол-во чисел 1 -50 4 2 -43 14 3 -30 28 4 0 7 5 10 1 6 12 6 7 15 18 8 30 11 9 40 11 10 49 0
2	lenArr = 40 xmin= -20 xmax= 20 NInt= 4 LGrInt={-20, -10, 0, 10 }	№ Лев.гр. Кол-во чисел 1 -20 15 2 -10 7 3 0 11 4 10 7
3	lenArr =1100 xmin=0 xmax=100 NInt=7 LGrInt={0, 5, 23, 56, 70, 75, 90}	№ Лев.гр. Кол-во чисел 1 0 48 2 5 211 3 23 360 4 56 149 5 70 56 6 75 155 7 90 121
4	lenArr =80 xmin=0	№ Лев.гр. Кол-во чисел 1 0 14

	xmax=10 NInt=5 LGrInt={ 0, 2, 4, 6, 8}	2 2 13 3 4 13 4 6 15 5 8 25
5	lenArr =16000 xmin=-8 xmax=8 NInt=6 LGrInt={ -8 -4 -2 0 3 6}	№ Лев.гр. Кол-во чисел 1 -8 3956 2 -4 1982 3 -2 2071 4 0 2983 5 3 3002 6 6 2006

Выводы.

Была реализована связь модуля на ЯВУ и модулей на Ассемблере.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ.

Название файла: lab6.cpp

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

extern "C" {
    void first(int* result, int* arr, int minX);
    void second(int* result, int lenResult, int* LGrInt, int nInt, int minX,
int maxX, int* output);
}

int main() {
    int lenArr;
    int minX, maxX;
    int nInt;

    srand(time(NULL));

    std::cout << "Size of array = 16 * N, input N: ";
    std::cin >> lenArr;

    if (lenArr > 16 * 1024) {
        std::cout << "\nError of size!\n";
        return 0;
    }

    std::cout << "\nInput Min X: ";
    std::cin >> minX;

    std::cout << "\nInput Max X: ";
    std::cin >> maxX;

    std::cout << "\nInput count of intervals (<= 24): ";
    std::cin >> nInt;

    if (nInt <= 0 || nInt > 24) {
        std::cout << "\nError of count intervals\n";
        return 0;
    }

    std::cout << "\nInput left borders: ";

    int* LGrInt = new int[nInt];

    for (int i = 0; i < nInt; i++) {
        std::cin >> LGrInt[i];
        if (LGrInt[i] > maxX || LGrInt[i] < minX) {
            std::cout << "\nBorder out of array!\n";
            return 0;
        }
    }

    int* arr = new int[lenArr];
    for (int j = 0; j < lenArr; j++)
        arr[j] = minX + rand() % (maxX - minX + 1);

    int* result = new int[abs(maxX - minX) + 1];
    for (int k = 0; k <= abs(maxX - minX); k++)
        result[k] = 0;

    int lenResult = abs(maxX - minX) + 1;

    //first(result, arr, minX);
```

```
    int* output = new int[nInt + 1];

    for (int k = 0; k <= nInt; k++)
        output[k] = 0;

    //second(result, lenResult, LGrInt, nInt, minX, maxX, output);

    std::ofstream file;
    file.open("./output.txt");

    for (int i = 1; i < nInt; i++) {
        std::cout << i << ")  ";
        file << i << ")  ";

        std::cout << "Border: " << LGrInt[i - 1] << "  ";
        file << "Border: " << LGrInt[i - 1] << "  ";

        int count = 0;
        for (int j = 0; j <= abs(maxX - minX); j++) {
            if (output[j] >= LGrInt[i - 1] && output[j] < LGrInt[i])
                count++;
        }

        std::cout << "count in interval: " << count << "\n";
        file << "count in interval: " << count << "\n";

    }

    std::cout << '\n';

    file.close();

    delete[] LGrInt;
    delete[] arr;
    delete[] result;
    delete[] output;

    return 0;
}
```

Название файла: first.asm

```
.686

.MODEL flat, C
.DATA
.CODE

PUBLIC C first
first PROC C res: dword, arr: dword, minx: dword

    push esi
    push edi
    push ebp

    mov edi, res
    mov esi, arr

myloop:

    mov ebx, [esi]
    sub ebx, minx
    mov ebp, [edi + 4*ebx]
    inc ebp
    mov [edi + 4*ebx], ebp
    add esi, 4

    loop myloop
```

```
pop ebp
pop edi
pop esi

    ret

first ENDP
End
```

Название файла: second.asm

```
.686


.MODEL flat, C

.DATA

.CODE


PUBLIC C second

second PROC C result: dword, lenresult: dword, leftborders: dword, nint:
dword, minx: dword, maxx: dword, output: dword


    push esi

    push edi

    push ebp


    mov esi, leftborders

    mov ecx, nint

    mov eax, 0


filoop:

    mov eax, [esi]

    sub eax, minx

    mov [esi], eax

    mov esi, 4


loop filoop


mov edi, result

mov ecx, nint

mov esi, leftborders

sub ebx, ebx
```

```
mov eax, [esi]

sloop:

    push ecx
    mov ecx, eax
    push esi
    mov esi, output

    tloop:

        mov eax, [edi]
        add [esi+4*ebx], eax
        add edi, 4

    loop tloop

    pop esi

    mov eax, [esi]
    add esi, 4
    sub eax, [esi]
    neg eax

    inc ebx
    pop ecx
loop sloop

mov esi, output
mov ecx, nint
sub eax, eax

foloop:

    add eax, [esi]
    add esi, 4
```

```
loop foloop

mov esi, output
sub eax, lenresult
neg eax
add [esi+4*ebx], eax

pop ebp
pop edi
pop esi

second ENDP
END
```