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 1

Студентка гр. 1383 Седова Э.А.

Преподаватель Ефремов М.А.

Санкт-Петербург
2022

Цель работы.

Научиться связывать язык Ассемблера и язык высокого уровня так, чтобы функции ассемблерного модуля вызывались из программы на ЯВУ. Написать программу построения частичного распределения попаданий псевдослучайных целых 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 целых чисел - 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]).

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)

Для бригад с нечетным номером: подпрограмма формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы реализуется в виде одного ассемблерного модуля, сразу формирующего требуемое распределение и возвращающего его в головную программу, написанную на ЯВУ;

Выполнение работы.

В начале программы подключаются необходимые библиотеки. Далее происходит считывание входных данных. Длина массива псевдослучайных целых чисел хранится в переменной *Numbers*, левая граница массива в переменной *left*, правая – в *right*, количество интервалов разбиения – в *NInt*, левые границы интервалов - в массиве *LGrInt*. Числа генерируются с помощью *rand()* и записываются в массив array. Далее управление переходит в ассемблерный модуль.

В ассемблерном модуле, в функции *func()*, реализован цикл по всем элементам массива чисел. Для каждого элемента в метке *next_border* производится поиск соответствующего ему интервала. Если текущее число меньше текущей левой границы происходит переход на метку *this_border*, где соответствующий результат увеличивается на единицу. Данный цикл длится пока не перебраны все элементы массива.

По завершении программы строки выводятся в консоль и записывается в файл ‘result.txt’.

Программный код см. в приложении А.

Тестирование

Результаты тестирования представлены на рисунках 1 и 2.

```
Длина массива псевдослучайных целых чисел:
12
Левая граница массива:
-1
Правая граница массива:
1
Количество интервалов разбиения: 3
Введите левую границу интервала[1]: -1
Введите левую границу интервала[2]: 0
Введите левую границу интервала[3]: 1
Массив: 1 1 0 0 1 0 -1 -1 0 1 1 1
Номер интервала:        Левая граница:  Количество чисел, попавших в интервал:
    1                   -1              2
    2                   0               4
    3                   1               6
```

Рисунок 1 – тест 1

```
Длина массива псевдослучайных целых чисел:
45
Левая граница массива:
-34
Правая граница массива:
67
Количество интервалов разбиения: 4
Введите левую границу интервала[1]: -23
Введите левую границу интервала[2]: -3
Введите левую границу интервала[3]: 5
Введите левую границу интервала[4]: 56
Массив: 7 -29 -24 48 61 -18 20 50 0 52 61 61 -9 65 33 49 3 -26 -1 -4 23 -16 -8 17 54 6 47 16 -2 -29 7 -34 49 -22 -1 -12
31 51 -33 -34 7 11 -26 5 -7
Номер интервала:        Левая граница:  Количество чисел, попавших в интервал:
    1                   -23             8
    2                   -3              5
    3                   5               20
    4                   56              4
```

Рисунок 2 – тест 2

Вывод.

В ходе выполнения работы научились связывать язык Ассемблера и язык высокого уровня так, чтобы функции ассемблерного модуля вызывались из программы на ЯВУ. Написана программа построения частичного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

Приложение А
Исходный код программы

Название файла:6lab.cpp

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

extern "C" void(func(int* array, int Numbers,
     int* LGrInt, int NInt, int* result));

int main() {
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
     std::ofstream Result;
    Result.open("result.txt", std::ios_base::out);

    int Numbers;
    std::cout << "Длина массива псевдослучайных целых чисел: \n";
    std::cin >> Numbers;
     if (Numbers <= 0 || Numbers > 16 * 1024) {
          printf("Неверная длина массива\n");
          return 1;
     }
    int* array = new int[Numbers];

    int left;
    std::cout << "Левая граница массива: \n";
    std::cin >> left;

    int right;
    std::cout << "Правая граница массива: \n";
    std::cin >> right;
     if (left > right) {
          printf("Неверный диапазон генерации чисел\n");
          return 1;
     }

     for (int i = 0; i < Numbers; ++i) {
          array[i] = rand() % (right - left + 1) + left;
     }

    int NInt;
    std::cout << "Количество интервалов разбиения: ";
    std::cin >> NInt;
     if (NInt <= 0) {
          printf("Неверное количество интервалов разбиения\n");
          return 1;
     }

     int* LGrInt = new int[NInt];
     int tmp;
     for (int i = 0; i < NInt; i++) {
```

```
            std::cout << "Введите левую границу интервала[" << i+1 <<
"]: ";
            std::cin >> tmp;
            LGrInt[i] = tmp;
            if (LGrInt[i] < left || LGrInt[i] > right ||
                  i > 0 && LGrInt[i] < LGrInt[i - 1] || LGrInt[i] ==
LGrInt[i - 1]) {
                  printf("Некорректное значение левой границы\n");
                  free(LGrInt);
                  return 1;
            }
      }

    int* result = new int[NInt] {0};
    func(array, Numbers, LGrInt, NInt, result);

      std::cout << "Массив: ";
      Result << "Массив: ";
      for (int i = 0; i < Numbers; ++i) {
            std::cout << array[i] << ' ';
            Result << array[i] << ' ';
      }
      std::cout << '\n';
      Result << '\n';
      std::cout << "Номер интервала:\t" << "Левая граница:\t" <<
"Количество чисел, попавших в интервал:\n";
      Result << "Номер интервала:\t" << "Левая граница:\t" <<
"Количество чисел, попавших в интервал:\n";
      for (int i = 0; i < NInt; ++i) {
            std::cout << "    " << i + 1 << "\t        \t" << LGrInt[i]
<< "\t\t" << result[i] << '\n';
            Result << "    " << i + 1 << "\t        \t" << LGrInt[i] <<
"\t\t" << result[i] << '\n';
      }

    delete[] array;
    delete[] LGrInt;
    delete[] result;
    Result.close();
    return 0;
}
```

Название файла:6lb.asm

```
.486
.MODEL FLAT
.CODE
public func
func proc PROC C array:dword,
Numbers:dword, LGrInt:dword, NInt:dword, result:dword

push eax
push ebx
push edi
push esi

mov esi, array
mov edi, LGrInt
```

```
mov eax, 0

start:
     mov ebx, 0 ;индекс текущего интервала
     next_border:
          cmp ebx, NInt
          jge this_border       ;если ebx>=NInt выходим из цикла
          push eax
          mov eax, [esi + 4 * eax]   ;кладем в eax элемент массива
array
          cmp eax, [edi + 4 * ebx]   ;сравниваем элемент с текущей
левой границей
          pop eax
          jl this_border  ;если eax<[edi + 4 * ebx] выходим из цикла
          inc ebx
          jmp next_border ;иначе переходим к следующему интервалу

     this_border:    ;когда вышли из цикла
          dec ebx
          cmp ebx, -1
          je next_num           ;если вышли за границу интервалов
          mov edi, result
          push eax
          mov eax, [edi + 4 * ebx]   ;помещаем в eax помещаем элемент
массива
                                                ;result с индексом ebx
          inc eax
          mov [edi + 4 * ebx], eax   ;устанавливаем полученное
значение
                                                ;обратно в массив
result
          pop eax
          mov edi,LGrInt

     next_num:
          inc eax         ;индекс следующего числа
          cmp eax, Numbers
          jg exit
jmp start       ;начинаем новую итерацию

exit:
pop esi
pop edi
pop ebx
pop eax

ret

func ENDP
END
```