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.

Студент гр. 9382 _____ Савельев И.С.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2020

Цель работы.

Организовать связи Ассемблера с ЯВУ на примере программы построения частотного распределение 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)

Ход выполнения.

В качестве ЯВУ использовался C++. С его помощью считывались все необходимые данные, после чего вызывался ассемблерный модуль, который обрабатывал входной массив и возвращал готовый результат в C++.

Тестирование.

Входные данные	Выходные данные
Введите длину массива: 10 Введите нижний интервал: 1 Введите верхний интервал: 10 Введите количество интервалов: 2 Введите 1 нижних границ интервалов: 5	Набор случайных чисел: 2 6 2 6 5 4 7 5 4 9 Номер интервала Левая граница Количество чисел 1 1 6 2 5 4
Введите длину массива: 50 Введите нижний интервал: 1 Введите верхний интервал: 50 Введите количество интервалов: 3 Введите 2 нижних границ интервалов: 0 Введенная граница 0 не входит в заданные промежутки! Введите еще раз: 10 25	Набор случайных чисел: 30 21 38 16 27 16 45 9 18 39 25 20 26 2 20 19 12 26 32 35 1 38 2 26 45 27 22 19 48 9 3 28 49 49 48 24 20 30 9 42 26 10 35 36 35 38 15 34 3 22 Номер интервала Левая граница Количество чисел 1 1 9 2 10 15 3 25 26

Вывод.

Была реализована программа организующая связь ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы.

Приложение А. Исходный код программы.

Файл main.cpp

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

extern "C" void INTERVAL_SORTING(int64_t* LGrInt, int64_t* result,
int64_t* array, int64_t NInt);

int64_t getRandomNumber(int64_t min, int64_t max) {
    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_int_distribution<int> dist(min, max);
    return dist(mt);
}

int main() {
    int64_t NInt = 0;
    int64_t Xmin = 0;
    int64_t Xmax = 0;
    int64_t count = 0;
    int64_t res;

    std::cout << "Введите длину массива: ";
    std::cin >> NInt;

    while (NInt > 16384) {
    std::cout << "Длинна больше допустимой,
введите заново: ";
    std::cin >> NInt;
    }

    std::cout << "Введите нижний интервал: ";
    std::cin >> Xmin;

    std::cout << "Введите верхний интервал: ";
    std::cin >> Xmax;

    while (Xmax <= Xmin) {
    std::cout << "Введен не коректный верхний
интервал, введите еще раз: " << '\n';
    std::cin >> Xmax;
    }

    std::cout << "Введите количество
интервалов: ";
    std::cin >> count;
```

```
          while (count > 24) {
          std::cout << "Введено не коректное число, введите еще раз: ";
          std::cin >> count;
          }

          int64_t *LGrInt = new int64_t[count];
          std::cout << "Введите " << count - 1 << " нижних границ интервалов: ";

          for (int64_t i = 0; i < count - 1; i++) {
          std::cin >> LGrInt[i];
          while (LGrInt[i] > Xmax || LGrInt[i] < Xmin) {
               std::cout << "Введеная граница " << LGrInt[i] << " не входит в заданные промежутки! Введите еще раз: ";
               std::cin >> LGrInt[i];
          }
          }

          LGrInt[count - 1] = Xmax;

          int64_t *array = new int64_t[NInt];

          for (int64_t i = 0; i < NInt; i++) {
          array[i] = getRandomNumber(Xmin, Xmax);
          }

          int64_t *result = new int64_t[count];
          for (int64_t i = 0; i < count; i++) {
          result[i] = 0;
          }

          INTERVAL_SORTING(LGrInt, result, array, NInt);

          std::ofstream out_file("result.txt");
          std::cout << "Набор случайных чисел: ";
          out_file << "Набор случайных чисел: ";
          for (int64_t i = 0; i < NInt; i++) {
          std::cout << array[i] << " ";
          out_file << array[i] << " ";
          }
          out_file << "\n";
          std::cout << "\n";
          std::cout << "\nНомер интервала\tЛевая граница\tКоличество чисел\n";
          out_file << "\nНомер интервала\tЛевая граница\tКоличество чисел\n";

          for (int64_t i = 0; i < count; i++) {
```

```
          if(i != 0) {
               res = LGrInt[i - 1];
          }
          else{
               res = Xmin;
          }
               out_file << "    " << i+1 << "\t\t     "    <<    res    <<
"\t\t\t    " << result[i] << "\n";
               std::cout << "        " << i+1 << "\t " << res << "\t\t
     " << result[i] << "\n";
          }

          delete result;
          delete array;
          delete LGrInt;

      }
```

Файл assem.s

```
      .intel_syntax noprefix
      .global INTERVAL_SORTING
      .text

      INTERVAL_SORTING:  # rdi : LGrInt,   rsi : result ,   rdx : array,
     rcx : NInt

          WokWork:
             mov  rax,  [rdx]      #  в  rax  л е ж и т  т е к у щ и й
э л е м е н т
           push rdx          #с о х р а н я е м  у к а з а т е л ь  н а
т е к у щ и й  э л е м е н т
           mov rdx, 0       # о б н у л я е м  у к а з а т е л ь

          Index_case:
           mov rbx, rdx   # в rbx л е ж и т  т е к у щ и й  и н д е к с
и з  LGrInt
           shl rbx, 3    # э т о т  и н д е к с  у м н о ж а е м  н а  8,
т . е .  к а ж д ы й  э л е м е н т  п о  8  б а й т
           cmp rax, [rdi + rbx]   # с р а в н и в а е м  т е к у щ и й
э л е м е н т  м а с с и в а  с  т е к у щ е й  г р а н и ц е й
          jg search_case                #    е с л и    э л е м е н т
м а с с и в а  б о л ь ш е  г р а н и ц ы
          jmp write_case

          search_case:
           inc rdx                 #    д л я    п е р е х о д а    к
с л е д у ю щ е м у  и н д е к с у  м а с с и в а  г р а н и ц
          jmp Index_case
```

```
        write_case:
         add rbx, rsi          # rbx указывает на индекс
LGrInt
         mov rax, [rbx]       # rax хранит индекс LGrInt
который надо ++
         inc rax                        #    увеличиваем
индекс на один
         mov [rbx], rax;                #       возвращаем
обратно
         pop rdx                                          #
восстанавливаем указатель на array
         add rdx, 8                     #       перемещаем
указатель на следующий элемент массива
чисел
         loop WokWork          # в цикле проходим по
всем элементам массива

        ret
```