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 построения частотного распределения псевдослучайных целых чисел в заданные интервалы.

Студент гр. 9383 _____ Ноздрин В.Я.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Научиться реализовывать связь Ассемблера и ЯВУ. Написать программу для 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 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:

Текстовый файл, строка которого содержит:

- номер интервала,
- левую границу интервала,
- количество псевдослучайных чисел, попавших в интервал. Количество строк равно числу интервалов разбиения.

Ход работы.

В ходе работы была написана программа на языке C++ и два ассемблерных модуля. Первый ассемблерный модуль реализует распределение по единичным отрезкам массива целых чисел. Второй ассемблерный модуль по этому распределению формирует второе распределение по заданным интервалам. Первое распределение строится с помощью циклов и записи в новый массив количества повторений каждого числа. Второе распределение сначала переносит границы интервалов в положительный промежуток, а затем сравнивает их с числами, полученными в первом ассемблерном модуле. Для связи модулей и C++ используется спецификатор extern.

Исходный код см. в приложении А.

Тестирование.

Данные:

```
NumRanDat 10
Xmin -10
Xmax 10
NInt 3
LGrInt -5 0 5
randDataArr 9 4 -1 5 -2 3 -4 -2 5 10
```

Результат работы:

```
result 1 0 0 0 0 0 0 1 0 2 1 0 0 0 1 1 2 0 0 0 1 1
result 2
    -10 -> 0
    -5  -> 4
     0  -> 4
     5  -> 2
```

Выводы.

Была разработана программа на языке C++, использующая внешние ассемблерные модули.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Текст файла main.cpp

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

#include <ctime>

extern "C" {
    void unit(int* result1, int* randDataArr, int NumRanDat, int Xmin);
     void final(int* result1, int result_len, int* LGrInt, int NInt, int Xmin, int Xmax,
int* result2);
}

void readParams(int*, int*, int*, int*);
void readLGrInt(int*, int);
void printResult1(int*);
void printResult2(int*, int*, int, int, int, int, std::ofstream);

int main() {
  int NumRanDat, Xmin, Xmax, NInt;
  int *LGrInt = nullptr;

  readParams(&NumRanDat, &Xmin, &Xmax, &NInt);
  readLGrInt(LGrInt, NInt);

  for (int i = 0; i < NInt; i++)
    if (LGrInt[i] > Xmax || LGrInt[i] < Xmin) {
      delete [] LGrInt;
      return 0;
    }

  int *randDataArr  = new int[NumRanDat];
  int result_len    = abs(Xmax-Xmin)+1;
  int *result1      = new int[result_len];
  int *result2      = new int[NInt+1];

  srand(time(NULL));
  for (int i = 0; i < NumRanDat; i++)
      randDataArr[i] = Xmin+rand()%(Xmax-Xmin+1);
  for (int i = 0; i < result_len; i++)
      result1[i] = 0;
```

```
  for (int i = 0; i < NInt+1; i++)
      result2[i] = 0;

  unit(result1, randDataArr, NumRanDat, Xmin);

  printResult1(result1);

  final(result1, result_len, LGrInt, NInt, Xmin, Xmax, NumRanDat, result2);

  std::ofstream output;
  file.open("output.txt");
  printResult2(result2, LGrInt, NInt, Xmin, Xmax, output);

  file.close();
  delete [] LGrInt;
  delete [] randDataArr;
  delete [] result1;
  delete [] result2;

  return 0;
}

void readParams(int *NumRanDat, int *Xmin, int *Xmax, int *NInt) {
  std::cout << "Enter NumRanDat\n";       std::cin >> &NumRanDat;
  if (&NumRanDat > 16*1024) { &NumRanDat = 16*1024; }
  std::cout << "Enter Xmin\n";            std::cin >> &Xmin;
  std::cout << "Enter Xmax\n";            std::cin >> &Xmax;
  std::cout << "Enter NInt\n";            std::cin >> &NInt;
  if (&NInt <= 0 || &NInt > 24) { &NInt = 2; }
}

void readLGrInt(int *LGrInt, int NInt){
  LGrInt = new int[NInt];
  std::cout << "Enter LGrInt randDataArray";
  for (int i = 0; i < NInt; i++)
    std::cin >> LGrInt[i];
}

void printResult1(int* result1){
  for (int j = 0; j < result_len; j++)
    std::cout << result1[j] << ' ';
  std::cout << '\n';
}

void  printResult2(int  *result2,  int  *LGrInt,  int  NInt,  int  Xmin,  int  Xmax,  int
NumRanDat, std::ofstream output){
  std::cout << "Result:\n";               file << "Result:\n";
  std::cout << "DataArray: ";             file << "DataArray: ";
```

```
  for (int i = 0; i < NumRanDat; i++) {
    std::cout << randDataArr[i] << ' ';   file << randDataArr[i] << ' ';
  }
  std::cout << '\n';                      file << '\n';

  for (int i = 1; i < NInt; i++) {
    std::cout << i << ") \t";
    file      << i << ") \t";
    std::cout << "Left Border: " << LGrInt[i - 1] << " ";
    file      << "Left Border: " << LGrInt[i - 1] << " ";
    int c = 0;
    for (int j = 0; j < result_len; j++) {
      if (i == NInt)
        if (result2[j] >= LGrInt[i - 1]) {
          c++;
          continue;
        }
      if (result2[j] >= LGrInt[i - 1] && result2[j] < LGrInt[i])
        c++;
    }
    std::cout << "\tcount in interval: " << c << "\n";
    file      << "\tcount in interval: " << c << "\n";
  }
}
```

Текст файла unit.asm

```
.686
.MODEL flat, C
.DATA
.CODE
PUBLIC C unit
unit PROC C result1: dword, randDataArr: dword, NumRanDat: dword, Xmin: dword
    push esi
    push edi
    push ebp

    mov edi, result1
    mov esi, randDataArr
    mov ecx, NumRanDat
    mov eax, Xmin

    for_loop:
        mov ebx, [esi]
        sub ebx, eax
        mov ebp, [edi + 4*ebx]
        inc ebp
```

```
        mov [edi + 4*ebx], ebp
        add esi, 4
    loop for_loop

    pop ebp
    pop edi
    pop esi

    ret

unit ENDP
END
```

Текст файла final.asm

```
.686
.MODEL flat, C
.DATA
.CODE
PUBLIC C final
final PROC C result1: dword, result_len: dword, LGrInt: dword, NInt: dword, Xmin: dword,
Xmax: dword, result2: dword

    push esi
    push edi
    push ebp

    mov esi, LGrInt
    mov edx, Xmin
    mov ecx, NInt
    mov eax, 0

    for_loop_1:
        mov eax, [esi]
        add eax, Xmax
        mov [esi], eax
        mov esi, 4
    loop for_loop_1

    mov edi, result1
    mov ecx, NInt
    mov esi, LGrInt
    sub ebx, ebx
    mov eax, [esi]

    for_loop_2:
        push ecx
```

```
        mov ecx, eax
        push esi
        mov esi, result2

        for_loop_3:
            mov eax, [edi]
            add [esi+4*ebx], eax
            add edi, 4
        loop for_loop_3

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

        inc ebx
        pop ecx
    loop for_loop_2

    mov esi, result2
    mov ecx, NInt
    sub eax, eax

    for_loop_4:
        add eax, [esi]
        add esi, 4
    loop for_loop_4

    mov esi, result2
    sub eax, result_len
    neg eax
    add [esi+4*ebx], eax

    pop ebp
    pop edi
    pop esi

final ENDP
END
```