

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 распределения попаданий псевдослучайных
целых чисел в заданные интервалы.

Студентка гр. 9382

Пя С.

Преподаватель

Ефремов М.А.

Санкт-Петербург

2020

Цель работы.

Изучить работу программы на ЯВУ с ассемблером, написать ассемблерный модуль.

Задание:

15 Вариант – нечетный

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 ($\leq 16K$, $K=1024$)
2. Диапазон изменения массива псевдослучайных целых чисел $[X_{\min}, X_{\max}]$, 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 $[X_{\min}, X_{\max}]$).

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:

1. 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;

Ход работы:

В качестве ЯВУ используется C++. Производится ввод необходимых данных, затем вызывается функция ассемблерная. В ней происходит обработка массива и выполняются необходимые действия для получения массива-ответа. В массиве-ответе хранится количество чисел, попавших в различные заданные диапазоны. Далее происходит вывод значений-ответов на экран и в файл.

Тестирование.

| Ввод данных | Вывод данных | | | | | | | | | | | | | | | | | | |
|--|---|---------------|-------------------|-------------------|---|----|---|---|----|---|---|----|---|---|----|---|---|----|---|
| Enter length of array: 5 | <table><tr><th>n_of_interval</th><th>l_brs</th><th>cnt_of_p- rm_n</th></tr><tr><td>1</td><td>10</td><td>0</td></tr><tr><td>2</td><td>15</td><td>0</td></tr><tr><td>3</td><td>20</td><td>1</td></tr><tr><td>4</td><td>30</td><td>0</td></tr><tr><td>5</td><td>40</td><td>4</td></tr></table> | n_of_interval | l_brs | cnt_of_p- rm_n | 1 | 10 | 0 | 2 | 15 | 0 | 3 | 20 | 1 | 4 | 30 | 0 | 5 | 40 | 4 |
| n_of_interval | | l_brs | cnt_of_p- rm_n | | | | | | | | | | | | | | | | |
| 1 | | 10 | 0 | | | | | | | | | | | | | | | | |
| 2 | | 15 | 0 | | | | | | | | | | | | | | | | |
| 3 | | 20 | 1 | | | | | | | | | | | | | | | | |
| 4 | | 30 | 0 | | | | | | | | | | | | | | | | |
| 5 | | 40 | 4 | | | | | | | | | | | | | | | | |
| Enter low range: 10 | | | | | | | | | | | | | | | | | | | |
| Enter high range: 100 | | | | | | | | | | | | | | | | | | | |
| Enter number of ranges(<= 24): 5 | | | | | | | | | | | | | | | | | | | |
| Enter 4 lower bounds of intervals: 5 | | | | | | | | | | | | | | | | | | | |
| Entered bound 5 are not included in the specified intervals! Enter again 101 | | | | | | | | | | | | | | | | | | | |
| Entered bound 101 are not included in the specified intervals! Enter again 4 | | | | | | | | | | | | | | | | | | | |
| Entered bound 4 are not included in the specified intervals! Enter again 15 | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | |
| Entered bound 14 are lower than previous! Enter again 20 | | | | | | | | | | | | | | | | | | | |

30

40

Generated pseudo-random numbers:

51 27 44 50 99

Выводы.

В результате выполнения лабораторной работы был разработан код, состоящий из ассемблерного модуля и остальной части на ЯВУ C++, который выводит частоту попадания псевдо-случайных чисел в определенные диапазоны.

Приложение.

Текст файла *main.cpp*

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdlib.h>
#include <iostream>
#include <fstream>
#include <ctime>
#include <random>

extern "C"
{
    void _form_array(int NumRanDat, int* arr, int* LGrInt, int* res_arr);
}

int main()
{
    int NumRanDat = 0;
    std::cout << "Enter length of array:\n";
    std::cin >> NumRanDat;
    if (NumRanDat > 16 * 1024) {
        std::cout << "Array is too long! It must not exceed 16*1024\n";
        return 0;
    }
    int Xmin = 0, Xmax = 0, NInt = 0;
    std::cout << "Enter low range:\n";
    std::cin >> Xmin;
    std::cout << "Enter high range:\n";
    std::cin >> Xmax;
    std::cout << "Enter number of ranges(<= 24): ";
    std::cin >> NInt;
    if (NInt > 24) {
        std::cout << "There are too many ranges! It must be less than or equal to 24\n";
        return 0;
    }
    int* LGrInt = new int[NInt]();
    std::cout << "Enter " << NInt - 1 << " lower bounds of intervals:\n";
    for (int i = 0; i < NInt - 1; i++) {
        std::cin >> LGrInt[i];
        while (LGrInt[i] < LGrInt[i - 1]) {
            std::cout << "Entered bound " << LGrInt[i] << " are lower than previous!
Enter again\n";
            std::cin >> LGrInt[i];
        }
    }
}
```

```

        while (LGrInt[i] < Xmin || LGrInt[i] > Xmax) {
            std::cout << "Entered bound " << LGrInt[i] << " are not included in the
specified intervals! Enter again\n";
            std::cin >> LGrInt[i];
        }
    }
    LGrInt[NInt - 1] = Xmax;
    int* arr = new int[NumRanDat]();
    for (int i = 0; i < NumRanDat; i++) {
        arr[i] = Xmin + rand() % (Xmax - Xmin);
    }
    int* res_arr = new int[NInt];
    for (int i = 0; i < NInt; i++)
        res_arr[i] = 0;
    _form_array(NumRanDat, arr, LGrInt, res_arr);
    std::ofstream file("res.txt");
    std::cout << "Generated pseudo-random numbers:\n";
    file << "Generated pseudo-random numbers:\n";
    for (int i = 0; i < NumRanDat; i++) {
        std::cout << arr[i] << " ";
        file << arr[i] << " ";
    }
    std::cout << "\n";
    file << "\n";
    std::cout << "\nn_of_interval\tl_brs\tcnt_of_p-rm_n\n";
    file << "\nn_of_interval\tleft_brs\tcnt_of_p-rm_n\n";
    for (int i = 0; i < NInt; i++) {
        int res = i != 0 ? LGrInt[i - 1] : Xmin;
        file << " " << i + 1 << "\t\t" << res << "\t\t" << res_arr[i] << "\n";
        std::cout << " " << i + 1 << "\t\t" << res << "\t\t" << res_arr[i] <<
"\n";
    }
}

```

Текст файла text.asm

```

.686
.MODEL FLAT, C
.STACK
.DATA
.CODE
_form_array PROC C NumRanDat:dword, arr:dword, LGrInt:dword, res_arr:dword

mov ecx,0 ;счетчик для прохода по массиву
mov ebx,[arr] ;входной массив
mov esi,[LGrInt] ;массив с левыми границами
mov edi,[res_arr] ;массив-ответ

fst_case:
    nop
    mov eax,[ebx] ;берем элемент входного массива
    push ebx ; сохраняем указатель на текущий элемент
    mov ebx,0 ; обнуляем указатель

snd_case:
    mov edx,ebx ; edx содержит текущий индекс массива границ
    shl edx,2 ; индекс умножаем на 4, так как каждый элемент по 4 байт
    cmp eax,[esi+edx] ; сравниваем текущий элемент с текущей левой границей
jg searching_case
jmp exepting_case

searching_case:
    inc ebx; инкрементируем, пока не найдем нужный интервал
jmp snd_case

```

```
exepthing_case:
    add edx,edi ;в массиве-ответе инкрементируем счетчик
    mov eax,[edx]
    inc eax
    mov [edx],eax;
    pop ebx ;забираем текущий элемент и ссылаемся на новый
    add ebx,4
    inc ecx ;инкрементируем индекс массива
    cmp ecx, NumRanDat
jl fst_case

ret
_form_array ENDP

END
```