# ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)»**

**Кафедра МО ЭВМ**

**ОТЧЁТ**

**по лабораторной работе № 6**

**по дисциплине «Организация ЭВМ и систем»**

**Тема: Организация связи Ассемблера с ЯВУ на примере программы построения псевдослучайных целых чисел в заданные интервалы.**

Студент гр. 1303 Бутыло Е.А.

Преподаватель Ефремов М.А.

Санкт-Петербург 2022

# Цель работы.

Рассмотреть способ организации связи Ассемблера с ЯВУ. Разработать программу построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

# Задание.

Вариант 1.

На языке высокого уровня (Pascal или С) генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих равномерное распределение. Необходимые датчики псевдослучайных чисел находятся в каталоге Tasks\RAND\_GEN (пpи его отсутствии программу датчика получить у пpеподавателя).

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.

* Длина массива псевдослучайных целых чисел — NumRanDat (*≤*

16*K, K* = 1024);

* Диапазон изменения массива псевдослучайных целых чисел

*Xmin, Xmax*, значения могут быть биполярные;

* Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел — NInt (*≤* 24);
* Массив левых границ интервалов разбиения LGrInt (должны

принадлежать итервалу [*Xmin, Xmax*]).

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.

Программа была написана с использованием двух ”модулей”: один написан на языке Си, в котором происходит считывание исходных данных и запись результатов, второй написан на языке Ассемблера — в нём происходит обработка данных. Для компиляции и линковки модулей была использована коллекция компиляторов GNU, команды объединены в Make- file.

В начале программы с помощью модуля на языке Си происходит считывание исходных данных, выделение памяти под массивы и их последующее заполнение согласно введённым данным; на каждом этапе проводится проверка данных — в случае некорректных данных программа выводит ошибку и завершается.

После вызывается процедура ассемблера processing\_intervals, которая принимает исходные данные (результирующий массив, массив чисел, массив левых границ интервалов, количество чисел, количество левых границ). Данная процедура для каждого числа из массива вызывает процедуру find\_interval\_index, которая в свою очередь, начиная с последнего интервала, проверяет, входит ли данное число в текущий интервал, — номер интервала записывается в регистр rax (0 в случае отсутствия подходящего интервала). Далее (в случае нахождения интервала), ячейка результирующего массива необходимого интервала увеличивается на 1.

В конце концов, массив с результатом средствами языка Си записывается в файл ”results.txt”.

# Выводы

Рассмотрены способы организации связи Ассемблера с ЯВУ, получены практические навыки в написании программы, использующей язык Си и Ассемблера. Разработана программа построения частотного распределения попаданий псевдослучайных целых чисел в заданные интервалы.

# ПРИЛОЖЕНИЕ A КОД ПРОГРАММ

Название файла: main.c

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
  
extern void processing\_intervals(int \*result\_array, int \*source\_array,  
 int \*borders\_array, int number, int borders\_number);  
  
int main() {  
 srand(time(NULL));  
  
 int numbers = 0;  
 int min\_border = 0;  
 int max\_border = 0;  
 int intervals = 0;  
  
 printf("Enter the data: count of numbers, max and min borders, count of intervals\n");  
 scanf("%d %d %d %d", &numbers, &min\_border, &max\_border, &intervals);  
  
 if (numbers <= 0 || numbers > 16 \* 1024) {  
 printf("Invalid count of numbers\n");  
 return 1;  
 } else if (min\_border >= max\_border) {  
 printf("Invalid max or min border\n");  
 return 1;  
 } else if (intervals <= 0 || intervals > 24) {  
 printf("Invalid count of intervals\n");  
 return 1;  
 }  
  
 int \*numbers\_array = malloc(numbers \* sizeof(int));  
 int \*intervals\_array = malloc(intervals \* sizeof(int));  
  
 printf("Enter left borders of intervals\n");  
 char c;  
 for (int i = 0; i < intervals; ++i) {  
 scanf("%d%c", &intervals\_array[i], &c);  
 if ((intervals\_array[i] < min\_border || intervals\_array[i] > max\_border) ||  
 (i > 0 && intervals\_array[i] <= intervals\_array[i - 1])) {  
 printf("Invalid left border!\n");  
 goto error\_free\_sources;  
 }  
 }  
  
 int rand\_max = max\_border - min\_border + 1;  
 for (int i = 0; i < numbers; ++i) {  
 numbers\_array[i] = min\_border + rand() % rand\_max;  
 }  
  
 int \*resultArray = calloc(intervals, sizeof(int));  
 processing\_intervals(resultArray, numbers\_array, intervals\_array, numbers, intervals);  
  
 FILE \*f = fopen("results.txt", "w");  
 if (!f) {  
 printf("Error creating file");  
 goto error\_free\_result;  
 }  
  
 fputs("Generated numbers:\n", f);  
 for (int i = 0; i < numbers; ++i) {  
 fprintf(f, "%d ", numbers\_array[i]);  
 }  
  
 fputs("\n\n", f);  
 fputs("Information processing results:\n", f);  
 for (int i = 0; i < intervals; ++i) {  
 fprintf(f, "Interval number: %d; Interval border: %d; Count of occurrences: %d.\n", i + 1, intervals\_array[i], resultArray[i]);  
 }  
  
 fclose(f);  
  
 return 0;  
  
 error\_free\_result:  
 free(resultArray);  
 error\_free\_sources:  
 free(numbers\_array);  
 free(intervals\_array);  
 return 1;  
};

Название файла: source.s

.global processing\_intervals  
  
processing\_intervals:  
 push rax  
  
get\_data\_loop:  
 lodsd  
  
 push rcx  
 mov rcx, r8  
  
 push rdi  
  
find\_interval\_index\_loop:  
 mov edi, [rdx + rcx \* 4 - 4]  
 cmp eax, edi  
 jge find\_interval\_index\_end  
 loop find\_interval\_index\_loop  
 xor rax, rax  
  
find\_interval\_index\_end:  
 mov rax, rcx  
  
 pop rdi  
  
 pop rcx  
  
 cmp rax, 0  
 je continue\_loop  
  
 inc dword ptr [rdi + rax \* 4 - 4]  
  
continue\_loop:  
 loop get\_data\_loop  
  
 pop rax  
 ret

Название файла: Makefile

all: main

main: main.o source.o

gcc main.o source.o -o main -z noexecstack

main.o: main.c

gcc -c main.c

source.o: source.s

as source.s -msyntax=intel -mnaked-reg -mmnemonic=intel -o source.o

clean:

rm -f \*.o main