

**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

Преподаватель

Токун Г.С.

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 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.**

- Длина массива псевдослучайных целых чисел — NumRanDat ( $\leq 16K$ ,  $K = 1024$ );
- Диапазон изменения массива псевдослучайных целых чисел  $X_{min}, X_{max}$ , значения могут быть биполярные;
- Количество интервалов, на которые разбивается диапазон изменения массива псевдослучайных целых чисел — NInt ( $\leq 24$ );
- Массив левых границ интервалов разбиения LGrInt (должны

принадлежать интервалу  $[X_{min}, X_{max}]$ ).

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 `process_data`, которая принимает исходные данные (результатирующий 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 А

### КОД ПРОГРАММ

Название файла: main.c

```
#include <stdio.h>
#include
<stdlib.h> #include
<time.h>

extern void process_data(int *result_array, int *source_array,
                        int *borders_array, int number, int borders_number);

int main() {
    srand(time(NULL))
    ;

    int n = 0;
    int x_min =
    0; int x_max
    = 0;

    int n_int = 0;

    printf("Enter: numbers' count, generation left border and right
           border, " "count of intervals\n");
    scanf("%d %d %d %d", &n, &x_min, &x_max, &n_int);

    if (n == 0 | n > 16 * 1024)
        { printf("Invalid numbers'
          count\n"); return 1;
        } else if (x_min == x_max) {
        printf("Invalid borders of number
          generation\n"); return 1;
        } else if (n_int == 0 | n_int >
        24) { printf("Invalid number of
          intervals\n"); return 1;
        }

    int *n_arr = malloc(n * sizeof(int));
    int *int_arr = malloc(n_int * sizeof(int));

    printf("Enter left borders of
    intervals\n"); char c;
    for (int i = 0; i <
        n_int;
        +i) {
        scanf("%d%c", &int_arr[i],
        &c);
        if ((int_arr[i] < x_min
            | int_arr[i] > x_max)
            | (i > 0 & int_arr[i]
            =
            int_arr[i - 1])) {
            printf("Invalid left
            border!\n"); goto
```

```
    error_free_sources;  
}  
}
```

```

int rand_max = x_max - x_min +
1; for (int i = 0; i < n;
        +i) {
    n_arr[i] = x_min + rand() % rand_max;
}

int *res_arr = calloc(n_int, sizeof(int));
process_data(res_arr, n_arr, int_arr, n,
n_int);

FILE *f = fopen("results.txt",
"w"); if (!f) {
    printf("Cannot create file to write
    results"); goto error_free_result;
}

fputs("Generated numbers:\n",
f); for (int i = 0; i < n;
        +i) {
    fprintf(f, "%d ", n_arr[i]);
}
fputs("\n\n", f);
fputs("Results:\n", f);
for (int i = 0; i < n_int; +i) {
    fprintf(f, "%d %d %d\n", i + 1, int_arr[i], res_arr[i]);
}

fclose(f);

return 0;

error_free_result
: free(res_arr);
error_free_sources
: free(n_arr);
free(int_arr);
return 1;
};

```

**Название файла:** lib.s

.global

process\_data #;

Input:

```

#; rdx > int *borders_array
#; rcx > int
borders_number #; rax >
int number

```

#; Output:

```

#; rax > int interval_index
+ 1 #; rax = 0 > not in
interval

```

```
find_interval_index:
```

```
find_interval_index_loop:  
    cmp    eax, [rdx + rcx * 4  
    - 4]
```



```

    jge
    find_interval_index_end
    loop
    find_interval_index_loop

find_interval_index_end:
    mov rax,

    rcx ret

;; Input:
;; rdi      > int *result_array
(qword)    ;; rsi      > int
*source_array (qword) ;; rdx  >
int *borders_array (qword) ;; rcx > int count
                                (dword)
;; r8  > int borders_number
(dword) process_data:
    push rax

process_data_loop:
    lodsd

    push rcx
    mov rcx,
    r8
    call
    find_interval_index
    pop rcx

    test rax, rax
    jz continue_loop

    inc dword ptr [rdi + rax * 4 -

4] continue_loop:
    loop process_data_loop

    pop rax
    ret

```

## Название файла: Makefile

```

all:
    main

main: main.o lib.o
    gcc main.o lib.o -o main -z noexecstack

main.o: main.c
    gcc -c main.c

lib.o: lib.s
    as lib.s -msyntax=intel -mnaked-reg -mmnemonic=intel -o lib.o

```

```
clean:
    rm -f *.o main
```