МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №6
по дисциплине «Организация систем и ЭВМ»
Тема «**Организация связи Ассемблера с ЯВУ на примере программы построения частотного распределение попаданий псевдослучайных целых чисел в заданные интервалы**»

Студентка гр. 1303 _____ Герасименко Я.Д.

Преподаватель _____ Ефремов М.А

Санкт-Петербург
2022

Цель работы.

Получить навыки программирования на языке Ассемблера. Разработать программу на ЯВУ с использованием языка Ассемблера, выполняющую построение частотного распределения попаданий псевдослучайных чисел в заданные интервалы.

Задание.

Вариант 23.

На языке С программируется ввод с клавиатуры и контроль исходных данных, а также генерируется массив псевдослучайных целых чисел, изменяющихся в заданном диапазоне и имеющих заданный закон распределения. Необходимые датчики псевдослучайных чисел находятся в каталоге RAND_GEN (при его отсутствии получить у преподавателя). Следует привести числа к целому виду с учетом диапазона изменения.

Далее должны вызываться 1 или 2 ассемблерных процедуры для формирования распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Ассемблерные процедуры должны вызываться как независимо скомпилированные модули. Передача параметров в процедуру должна выполняться через кадр стека.

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.

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.

В ходе выполнения работы написаны два файла – main.c и lab6.s – модуль, выполняющий обработку данных, написанный на языке Ассемблера. Также был создан Makefile.

В файле main.c в функции main() происходит считывание данных, проверка на корректность введенных данных, передача данных в ассемблерный модуль для их обработки. Полученный результат выводится в файл(res.txt) и в консолль средствами ЯВУ. Генерация чисел из заданного промежутка происходит с помощью функции rand().

В ассемблерном модуле обработка происходит следующим образом:

С помощью lodsd из массива сгенерированных чисел загружается двойное слово в регистр eax, далее, значение, хранящееся в eax сравнивается поочередно со значениями левых границ(начиная с последней), если это значение оказывается больше чем текущая левая граница, то происходит переход на метку end_find(интервал найден), иначе же сравнение происходит со следующей левой границей. Когда все границы перебраны, то берется следующий символ и для него происходит проверка. Когда перебраны все символы, то происходит выход из ассемблерного модуля.

В метке end_find значение, хранящееся в массиве res в конкретной ячейке увеличивается на 1, таким образом происходит подсчет. Измененный массив res теперь хранит результат выполнения работы.

Тестирование.

```
corvussharp@corvussharp-RedmiBook-16:~/Рабочий стол/lab6$ make
gcc -c main.c -o main.o
gcc main.o lab6.o -o main -z noexecstack
corvussharp@corvussharp-RedmiBook-16:~/Рабочий стол/lab6$ ./main
Сколько чисел будет?
10
диапазон генерации чисел:
0
10
Сколько будет интервалов разбиения?
3
Введите массив левых границ
1 2 3
1       1       2
2       2       1
3       3       6
corvussharp@corvussharp-RedmiBook-16:~/Рабочий стол/lab6$
```

Вывод.

В результате лабораторной работы была написана программа, корректно выполняющая формирование распределения количества попаданий псевдослучайных целых чисел в заданные интервалы. Рассмотрен способ организации связи Ассемблера с ЯВУ.

ПРИЛОЖЕНИЕ А

Название файла main.c.

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

extern void function(int* res_arr, int* nums_arr
                , int k_inter, int count_N, int* left_arr);

int main() {



    puts("Сколько чисел будет?");
    int countN;
    scanf("%d", &countN);
    if (countN <= 0 || countN > 16 * 1024) {
        puts("Что то не так ввели");
        return 1;
    }



    puts("диапазон генерации чисел: ");
    int d_l, d_r;
    scanf("%d %d", &d_l, &d_r);
    if (d_l > d_r) {
        puts("Что то не так с диапозоном");
        return 1;
    }



    puts("Сколько будет интервалов разбиения?");
    int k_inter;
    scanf("%d", &k_inter);
    if (k_inter <= 0) {
        puts("Нужно больше *Золота* интервалов");
        return 1;
    }



    puts("Введите массив левых границ");
    int* arr = malloc(sizeof(int) * k_inter);
    for (int i = 0; i < k_inter; i++) {
        scanf("%d", &arr[i]);
        if (arr[i] < d_l || arr[i] > d_r ||
        i > 0 && arr[i] < arr[i - 1]) {
            puts("Некорректное значение левой границы");
```

```
            free(arr);
            return 1;
        }
    }
    int* nums = malloc(sizeof(int) * countN);

    for (int i = 0; i < countN; i++) {
        nums[i] = rand() % (d_r - d_l + 1) + d_l;
    }
    int* res = calloc(k_inter, sizeof(int));

    function(res, nums, k_inter, countN, arr);

    FILE* f = fopen("res.txt", "w");
    for(int i=0; i < countN; i++){
        fprintf(f, "%d ", nums[i]);
    }
    fprintf(f, "\n\n");
    for (int i = 0; i < k_inter; i++) {
        fprintf(f, "%d\t%d\t%d\n", i + 1, arr[i], res[i]);
    }
    for (int i = 0; i < k_inter; i++) {
        printf("%d\t%d\t%d\n", i + 1, arr[i], res[i]);
    }
    fclose(f);
    return 0;
}
```

Название файла lab6.s.

```
.global function
#rdi - int *res_arr, rsi - int *nums_arr, rdx - int k_inter, rcx
- int count_N, r8 - int *left_arrarr
function:
     lodsd
     push rcx
     mov rcx, rdx
f_interval:
     cmp eax, [r8+rcx*4-4]# сравниваем с крайней правой границой
     jge end_find
     loop f_interval
     jmp exit
end_find:
     inc dword ptr [rdi+rcx*4-4]
exit:
     pop rcx
     loop function
     ret
```

Название файла Makefile

```
all: main

main: main.o lab6.o
```

```
	gcc main.o lab6.o -o main -z noexecstack
main.o: main.c
	gcc -c main.c -o main.o
lab6.o: lab6.s
	as  lab6.s  -mmnemonic=intel  -msyntax=intel  -mnaked-reg  -o
lab6.o

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