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 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 (≤ 24);
- Массив левых границ интервалов разбиения LGrInt (должны

принадлежать итервалу [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.

Написана программа состоящая из двух ”модулей”: один написан на языке Си, в нём считываются исходные данные, а также записываются результаты выполнения программы, второй написан на языке Ассемблера — здесь происходит обработка данных. Для компиляции и линковки модулей была использована коллекция компиляторов GNU, команды объединены в Make- file.

С помощью модуля написанного на ЯВУ Си считываются исходные данные, также выделяется память для их хранения и непосредственно её заполнение; при неверном формате входных данных ошибка обрабатывается и выводится соответствующая ошибка, а программа 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.

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 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
```