Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет Информационных технологий и программирования

Дополнительные задания.

Выполнила: Авакян Карина Артуровна

Проверил: Андреев Николай Владимирович.

Санкт-Петербург

2022 г.

Задание 1.

Разработать программу на языке С. Программа должна считать из файла целые числа типа int, количество чисел до 1000. Отсортировать полученный набор по возрастанию. Записать полученный набор в файл. Сортировка должна быть реализована в виде ассемблерной вставки.

Задание 2.

Разработать программу на языке С. Программа должна считать из файла целые числа типа int, количество чисел от 100 до 1000. Найти первые 10 наибольших значений. Записать эти значения в файл. Поиск первых 10 наибольших значений реализовать в виде ассемблерной вставки.

Файл sort.c*. Реализуем сортировку

```
#include "sort.h"
#include <stdio.h>
void sort(int* arr, int count)
{
   __asm {
      //void sort(int* arr, int count)
{
   __asm {
      //загрузка адреса массива arr в регистр ecx
      mov ecx, dword ptr [arr]
      // Инициализация переменных i, j, и tmp. Регистр esp уменьшается на 12 байт, чтобы выделить место для двух целых чисел и указателя на стеке.
      mov dword ptr [esp], 0
      sub esp, 12
            // Проверка условия цикла while (i < count). Загрузка значения i из стека в регистр eax, сравнение с count и переход к метке loop1_end, если i >= count.
      //while (i < count) {
   loop1_start:
      mov eax, dword ptr [esp + 12]//загружаем знач i в регистр eax, которое лежит по адресу exp+12
      cmp eax, dword ptr [count] // конструкция для сравнения
      jge loop1_end // конструкция для завершения цикла, выполняется следующий цикл
      //j = i + 1;
      mov eax, dword ptr [esp + 12]
      add eax, 1                    // Увеличение переменной j на 1.
      mov dword ptr [esp + 8], eax
      //while (j < count) {
   loop2_start:
      mov eax, dword ptr [esp + 8] //Загрузка значения j из стека в регистр eax
      cmp eax, dword ptr [count]   //сравнение с count   и переход к метке loop2_end, если j >= count.
      jge loop2_end
      //if (arr[i] > arr[j]) {
      mov eax, dword ptr [esp + 12]  // загружаем знач i в регистр eax,
      mov edx, dword ptr [esp + 8]   // Загрузка значения j из стека в регистр edx
      mov eax, dword ptr [ecx + eax * 4] // загружает значение из ячейки памяти, адрес которой вычисляется как [ecx + eax * 4], в регистр eax.
      cmp eax, dword ptr [ecx + edx * 4] // сравнение и переход к метке if_end, если arr[i] <= arr[j].
      jle if_end
      //tmp = arr[i];
      mov dword ptr [esp + 4], eax

 //arr[i] = arr[j]; //Здесь происходит обмен значениями между элементами массива с индексами i и j. Значение элемента массива с индексом j записывается в элемент массива с индексом i.
      mov eax, dword ptr [esp + 12] // загружаем знач i в регистр eax,
      mov edx, dword ptr [esp + 8] // Загрузка значения j из стека в регистр edx
      mov edx, dword ptr [ecx + edx * 4] // загружает значение из ячейки памяти, адрес которой вычисляется как [ecx + edx * 4], в регистр edx.
      mov dword ptr [ecx + eax * 4], edx // записывает значение из регистра edx в элемент массива, находящийся по индексу eax.
      //arr[j] = tmp; //Значение временной переменной tmp записывается в элемент массива с индексом j.
```

```
        mov eax, dword ptr [esp + 8] // загружаем знач i в регистр eax,
        mov edx, dword ptr [esp + 4] // Загрузка значения j из стека в регистр edx
        mov dword ptr [ecx + eax * 4], edx // // записывает значение из регистра edx в элемент массива, находящийся по индексу eax.
        //}
    if_end:
        //Увеличение переменной j на 1.
        mov eax, dword ptr [esp + 8]
        add eax, 1
        //}
        jmp loop2_start // Переход к метке loop2_start где начинается следующая итерация внутреннего цикла.
    loop2_end:
        // Увеличение переменной i на 1.
        mov eax, dword ptr [esp + 12]
        add eax, 1
        mov dword ptr [esp + 12], eax //
        //}
        jmp loop1_start// Переход к метке loop1_start для продолжения итерации внешнего цикла.
    loop1_end: //конец первого цикла
        add esp, 12
    }
}
        mov ecx, dword ptr [arr]
        //int i = 0, j, tmp;
        mov dword ptr [esp], 0
        sub esp, 12
        //while (i < count) {
    loop1_start:
        mov eax, dword ptr [esp + 12]
        cmp eax, dword ptr [count]
        jge loop1_end
        //j = i + 1;
        mov eax, dword ptr [esp + 12]
        add eax, 1
        mov dword ptr [esp + 8], eax
        //while (j < count) {
    loop2_start:
        mov eax, dword ptr [esp + 8]
        cmp eax, dword ptr [count]
        jge loop2_end
        //if (arr[i] > arr[j]) {
        mov eax, dword ptr [esp + 12]
        mov edx, dword ptr [esp + 8]
        mov eax, dword ptr [ecx + eax * 4]
        cmp eax, dword ptr [ecx + edx * 4]
        jle if_end
        //tmp = arr[i];
        mov dword ptr [esp + 4], eax
        //arr[i] = arr[j];
        mov eax, dword ptr [esp + 12]
        mov edx, dword ptr [esp + 8]
        mov edx, dword ptr [ecx + edx * 4]
        mov dword ptr [ecx + eax * 4], edx
        //arr[j] = tmp;
        mov eax, dword ptr [esp + 8]
        mov edx, dword ptr [esp + 4]
        mov dword ptr [ecx + eax * 4], edx
        //}
    if_end:
        //++j;
        mov eax, dword ptr [esp + 8]
        add eax, 1
        mov dword ptr [esp + 8], eax
        //}
        jmp loop2_start
    loop2_end:
```

```
        //++i;
        mov eax, dword ptr [esp + 12]
        add eax, 1
        mov dword ptr [esp + 12], eax
        //}
        jmp loop1_start
    loop1_end:
        add esp, 12
    }
}
void findMax10(int* arr, int count, int* res) {
    __asm {
        //sort(arr, count);
//Загружаev параметры функции sort (адрес массива arr и количество элементов count), и вызываev функцию sort для сортировки массива arr.
        mov eax, dword ptr [count]
        push eax
        mov ecx ,dword ptr [arr]
        push ecx
        call sort
        add esp, 8
        //int i = 0;// выделяем память для i
        mov dword ptr [esp], 0
        sub esp, 4
        //while (i < 10) {
    for_start:
        mov eax, dword ptr [esp + 4]
        cmp eax, 10 // Сравниваем значение переменной i с 10. IF i >= 10, то цикл завершается и переходим к метке for_end.
        jge for_end
        //res[i] = arr[count - i - 1]; // Вычисляет индекс элемента массива arr, соответствующего текущему значению переменной i. Элемент с этим индексом сохраняется в локальную переменную eax, а затем копируется в массив res с помощью инструкции mov.
        mov eax, dword ptr [count]
        mov ecx, dword ptr [esp + 4]
        sub eax, ecx
        sub eax, 1
        mov edx, dword ptr [arr]
        mov eax, dword ptr [edx + eax * 4]
        mov edx, dword ptr [res]
        mov dword ptr [edx + ecx * 4], eax
        //++i;
        mov eax, dword ptr [esp + 4]
        add eax, 1
        mov dword ptr [esp + 4], eax
        //}
        jmp for_start
    for_end:
        add esp, 4
    }
}
```

Main.c

// Показано использование функций sort и findMax10, реализована работа с текстовыми файлами

```
#include "sort.h"
#include <assert.h>
#include <stdio.h>

void test();

void taskSort() {
```

```
    int data[1000];
    int size = 0;
    int i;
    FILE* F = fopen("in1.txt", "r");
    assert(F != NULL);
    while (fscanf(F, "%d", &data[size]) == 1) {
        ++size;
    }
    fclose(F);

    sort(data, size);

    F = fopen("out1.txt", "w");
    assert(F != NULL);
    for (i = 0; i < size; ++i) {
        fprintf(F, "%d ", data[i]);
    }
    fclose(F);
}

void taskMax10()
{
    int data[1000];
    int maxVals[10];
    int size = 0;
    int i;
    FILE* F = fopen("in2.txt", "r");
    assert(F != NULL);
    while (fscanf(F, "%d", &data[size]) == 1) {
        ++size;
    }
    fclose(F);

    findMax10(data, size, maxVals);

    F = fopen("out2.txt", "w");
    assert(F != NULL);
    for (i = 0; i < 10; ++i) {
        fprintf(F, "%d ", maxVals[i]);
    }
    fclose(F);
}

int main() {
    test();
    taskSort();
    taskMax10();
    return 0;
}
```