Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №4**

«Обработка массивов. Числа Фибоначчи.»

Выполнил:

студент группы 213 Файтельсон А.А.

Проверил:

доцент кафедры ПОиАИС Кривонос А.В.

Курск, 2023

**Цели работы:**

1) изучение принципов функционирования памяти и микропроцессора компьютера при выполнении операций над массивами данных;

2) приобретение навыков использования команд ассемблера, связанных с обработкой массивов;

3) получение представления об особенностях обработки данных, команд и режимах доступа к данным при обработке массивов.

**Листинг программы**

data segment ; Начало сегмента данных

fibonacci dw 18 dup(?) ; Создаем массив из 18 элементов, каждый занимает 2 байта, значения изначально не инициализированы

min\_odd dw 0 ; Переменная для хранения минимального нечетного элемента

max\_even dw 0 ; Переменная для хранения максимального четного элемента

data ends ; Конец сегмента данных

code segment ; Начало сегмента кода

start:

mov ax, data ; Загружаем адрес сегмента данных в AX

mov ds, ax ; Устанавливаем сегмент данных DS

; Вычисление первых 18 чисел Фибоначчи и их запись в массив

mov cx, 16 ; Устанавливаем счетчик на 16 итераций, т.к. первые два числа уже заданы

mov si, 0 ; Начальный индекс массива fibonacci (указывает на текущий элемент)

mov ax, 0 ; Первое число Фибоначчи

mov [fibonacci + si], ax ; Сохраняем первое число в массив

add si, 2 ; Переходим к следующему элементу массива (увеличиваем индекс на 2 байта)

mov ax, 1 ; Второе число Фибоначчи

mov [fibonacci + si], ax ; Сохраняем второе число в массив

add si, 2 ; Переходим к следующему элементу массива

; Генерация оставшихся чисел Фибоначчи

generate\_fib:

mov bx, [fibonacci + si - 4] ; Загружаем F(n-2) из массива (смещение на -4 байта)

add bx, [fibonacci + si - 2] ; Добавляем F(n-1) к F(n-2)

mov [fibonacci + si], bx ; Сохраняем F(n) в массив

add si, 2 ; Переходим к следующему элементу массива

loop generate\_fib ; Повторяем цикл, пока счетчик CX не достигнет нуля

; Поиск минимального нечетного элемента во второй строке массива

mov cx, 6 ; Количество элементов во второй строке (6 столбцов)

mov si, 2 \* 6 ; Начальный индекс второй строки (первый элемент второй строки имеет индекс 6)

mov bx, 0FFFFh ; Устанавливаем начальное значение минимального элемента как максимально возможное (65535)

find\_min\_odd:

mov ax, [fibonacci + si] ; Загружаем текущий элемент массива

test ax, 1 ; Проверяем, является ли число нечетным (побитовое И с 1)

jz next\_odd ; Если четное, пропускаем обработку и переходим к следующему элементу

cmp ax, bx ; Сравниваем текущий элемент с текущим минимальным

jae next\_odd ; Если больше или равно минимальному, пропускаем обработку

mov bx, ax ; Если меньше, обновляем значение минимального

next\_odd:

add si, 2 ; Переходим к следующему элементу строки

loop find\_min\_odd ; Повторяем цикл для всех элементов строки

mov [min\_odd], bx ; Сохраняем минимальное нечетное значение в min\_odd

; Поиск максимального четного элемента в четвертом столбце

mov cx, 3 ; Количество строк (всего 3 строки)

mov si, 2 \* 3 ; Начальный индекс четвертого столбца (индекс 3)

mov bx, 0 ; Устанавливаем начальное значение максимального элемента как минимально возможное (0)

find\_max\_even:

mov ax, [fibonacci + si] ; Загружаем текущий элемент массива

test ax, 1 ; Проверяем, является ли число четным

jnz next\_even ; Если нечетное, пропускаем обработку и переходим к следующему элементу

cmp ax, bx ; Сравниваем текущий элемент с текущим максимальным

jbe next\_even ; Если меньше или равно максимальному, пропускаем обработку

mov bx, ax ; Если больше, обновляем значение максимального

next\_even:

add si, 2 \* 6 ; Переходим к следующей строке в том же столбце

loop find\_max\_even ; Повторяем цикл для всех строк

mov [max\_even], bx ; Сохраняем максимальное четное значение в max\_even

; Завершение программы

mov ax, 4C00h ; Код завершения программы

int 21h ; Вызов прерывания для завершения программы

code ends ; Конец сегмента кода

end start ; Конец программы, точка входа — метка start

**Вывод:** изучили принципы функционирования памяти и микропроцессора компьютера при выполнении операций над массивами данных;