**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Кубанский Государственный Университет**

Кафедра информационных технологий

ОТЧЕТ

о выполнении лабораторной работы №3

по дисциплине «Системы реального времени»

Выполнил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ст. гр. 44 Иванов А.А.\_\_\_\_\_\_\_

Проверил:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

доц. каф. ИТ Полетайкин А.Н.

Краснодар

2024

**Организация обработки числовых одномерных массивов**

Цель: изучение команд организации циклов и способов косвенной адресации данных памяти в микропроцессоре *i486*; приобретение практических навыков составления программ обработки одномерных массивов, освоение методов анализа трудоемкости и ресурсной сложности алгоритмов обработки одномерных числовых массивов.

**Задание**

1. На основании индивидуального задания составить программу для обработки элементов одномерного массива. Длина элементов исходного массива равна DW. Значения элементов исходного массива задать в сегменте данных. Длину элементов результирующего массива, если он необходим, выбрать самостоятельно.

2. Получить загрузочный модуль и протестировать выполнение программы.

3. Выполнить расчет времени выполнения программы.

Индивидуальные условия лабораторной работы представлены в таблице (Таблица 1):

|  |  |
| --- | --- |
| № варианта | Задание на обработку |
| 10 | Дан массив A[10]. Найти минимальный элемент Amin и рассчитать его факториал (Amin!) |

Таблица 1 – индивидуальные условия лабораторной работы

**Ход работы**

*1. На основании индивидуального задания составить программу для обработки элементов одномерного массива. Длина элементов исходного массива равна DW. Значения элементов исходного массива задать в сегменте данных. Длину элементов результирующего массива, если он необходим, выбрать самостоятельно.*

Программа, реализующая заданную задачу по обработке элементов одномерного массива, представлена ниже:  
.486

.model flat

.data

A DW 10, 20, 30, 40, 5, 6, 7, 8, 9, 10

A\_min DW ?

A\_cur DW ?

A\_min\_fact\_star DW -1

A\_min\_fact\_mlad DW -1

.code

\_start:

;нахождение минимального элемента

mov eax, offset A

mov ebx, 0

mov dx, [eax+ebx]

mov A\_min, dx

xor ecx, ecx

mov cx, 10

tipa\_for:

mov dx, [eax+ebx]

mov A\_cur, dx

mov sp, A\_cur

mov bp, A\_min

cmp sp, bp

jge skip

mov dx, A\_cur

mov A\_min, dx

skip:

add ebx, 2

loop tipa\_for

;проверка минимального на минус (факториал отрицательных чисел не определён) и подсчёт факториала

xor eax, eax

mov sp, 0

mov bp, A\_min

cmp sp, bp

jge minus

mov A\_min\_fact\_star, 0

mov sp, 1

mov A\_min\_fact\_mlad, sp

mov cx, A\_min

opiat\_for:

mov ax, A\_min\_fact\_mlad

mov bx, A\_min\_fact\_star

mov dx, A\_min

mul dx

mov A\_min\_fact\_mlad, ax

mov A\_min\_fact\_star, dx

mov dx, A\_min

dec dx

mov A\_min, dx

loop opiat\_for

minus:

ret

end \_start

Вышеописанная программа работает следующим образом:

1. Определяет начало массива, записывает первый элемент массива как минимальный, устанавливает кол-во итерационных шагов цикла равное длине массива;

2. Поочерёдно перебирает элементы массива с использованием косвенной адресации (базовая + индекс), сравнивая текущий элемент с минимальным и заменяя минимальный текущим, если он меньше минимального;

3. После определения минимального элемента проверяет его на наличие минуса (т.к. факториал отрицательных чисел не определён, то программа завершается, если полученный минимальный элемент меньше 0);

4. Если минимальный элемент не меньше 0, то значение факториала становится равно 1 (для дальнейшего умножения), устанавливается кол-во итераций цикла умножения и уменьшения минимального элемента на 1 (эта величина = самому минимальному элементу);

5. После завершения цикла умножения программа заканчивает свою работу.

Вышеописанная программа с данным массивом A = 10, 20, 30, 40, 5, 6, 7, 8, 9, 10 определяет минимальный элемент как 5, а затем считает для него факториал – 120 (78 в шестнадцатеричной).

*2. Получить загрузочный модуль и протестировать выполнение программы.*

На рисунках (Рисунок 1, Рисунок 2, Рисунок 3) представлен ход выполнения программы:

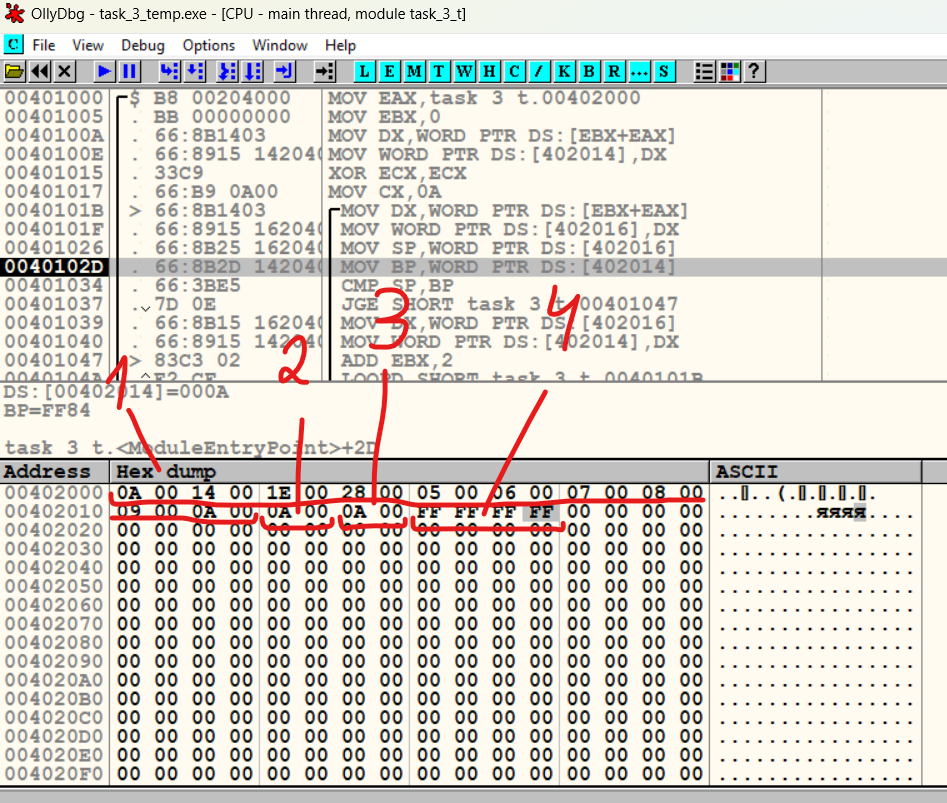


Рисунок 1 – старт работы программы

Где:

1 – массив чисел формата DW (10, 20, 30, 40, 5, 6, 7, 8, 9, 10);

2 – минимальный элемент на данной итерации (10);

3 – текущий элемент на данной итерации (10);

4 – значение факториала минимального элемента (на старте работы программы = -1).

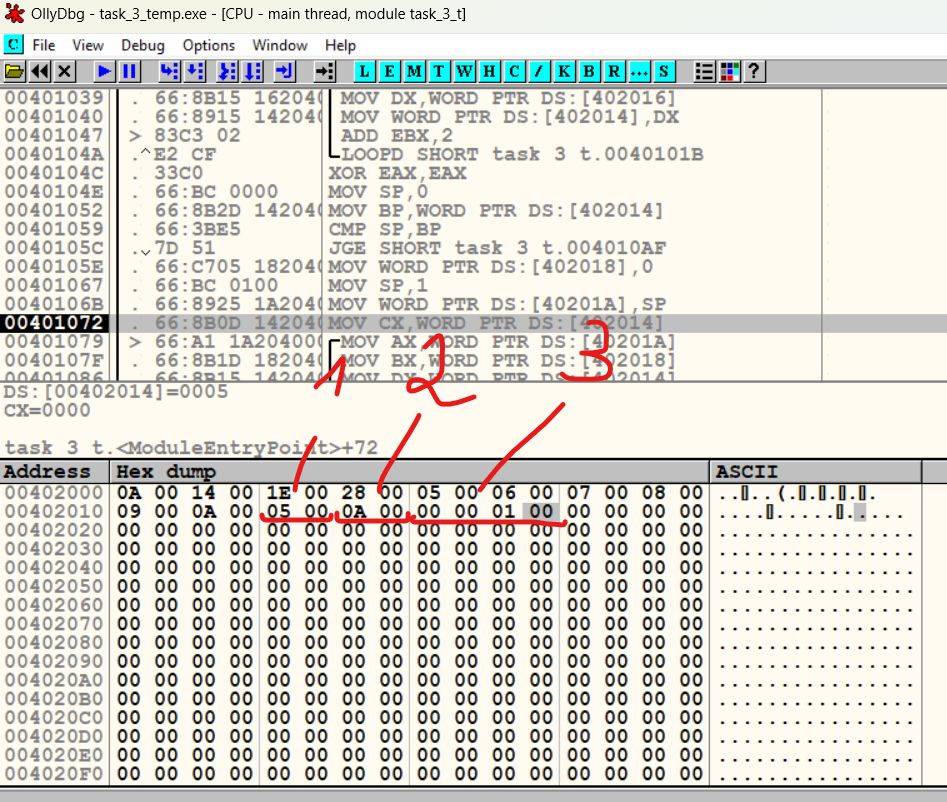


Рисунок 2 – промежуточный этап работы программы

Где:

1 – минимальный элемент массива после полного перебора (5);

2 – текущий минимальный элемент после полного перебора (10);

3 – прошла проверка, минимальный элемент > 0 и значение факториала становится = 1 для дальнейшего умножения.

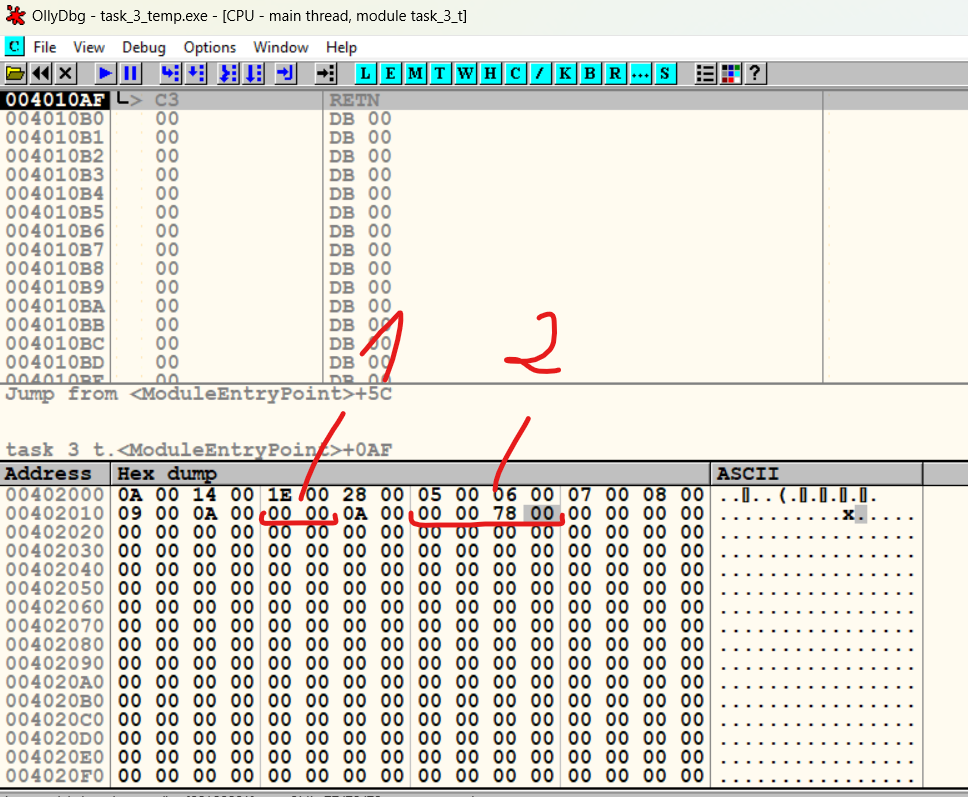


Рисунок 3 – окончание работы программы

Где:

1 – минимальный элемент стал 0, т.е. в цикле он постепенно уменьшался на 1 для подсчёта факториала;

2 – итоговое значение факториала минимального элемента (78 в шестнадцатеричной = 120 в десятичной).

*3. Выполнить расчет времени выполнения программы.*

Ниже представлен подсчёт времени выполнения программы:

.486

.model flat

.data

A DW 10, 20, 30, 40, 5, 6, 7, 8, 9, 10

A\_min DW ?

A\_cur DW ?

A\_min\_fact\_star DW -1

A\_min\_fact\_mlad DW -1

.code

\_start:

;нахождение минимального элемента

mov eax, offset A 8(12)+EA=24

mov ebx, 0 8(12)+ЕА=18

mov dx, [eax+ebx] 8(12)+ЕА=24

mov A\_min, dx 9(13)+ЕА=19

xor ecx, ecx 3

mov cx, 10 4

tipa\_for:

mov dx, [eax+ebx] 8(12)+ЕА=24

mov A\_cur, dx 9(13)+ЕА=19

mov sp, A\_cur 8(12)+ЕА=18

mov bp, A\_min 8(12)+ЕА=18

cmp sp, bp 3

jge skip 16

mov dx, A\_cur 8(12)+ЕА=18

mov A\_min, dx 9(13)+ЕА=19

skip:

add ebx, 2 4

loop tipa\_for 17

;проверка минимального на минус (факториал отрицательных чисел не определён) и подсчёт факториала

xor eax, eax 3

mov sp, 0 8(12)+ЕА=18

mov bp, A\_min 8(12)+ЕА=18

cmp sp, bp 3

jge minus 16

mov A\_min\_fact\_star, 0 10(14)+ЕА=22

mov sp, 1 4

mov A\_min\_fact\_mlad, sp 9(13)+ЕА=19

mov cx, A\_min 8(12)+ЕА=18

opiat\_for:

mov ax, A\_min\_fact\_mlad 8(12)+ЕА=18

mov bx, A\_min\_fact\_star 8(12)+ЕА=18

mov dx, A\_min 8(12)+ЕА=18

mul dx 118-133

mov A\_min\_fact\_mlad, ax 9(13)+ЕА=19

mov A\_min\_fact\_star, dx 9(13)+ЕА=19

mov dx, A\_min 8(12)+ЕА=18

dec dx 2

mov A\_min, dx 9(13)+ЕА=19

loop opiat\_for 17

minus:

ret 20

end \_start

Итоговое время выполнения программы: 92+156\*10+121+281\*5+20=3198 тактов.

Тактовая частота вычислительного процессора (12th Gen Intel(R) Core(TM) i5-12500H) = 2.50 ГГЦ.

Тогда в секундах это равно: 

А в наносекундах это равно: 