|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе №** 4

**Дисциплина: Машинно-зависимые языки и основы компиляции**

**Название лабораторной работы:** Обработка массивов и матриц

Студент гр. ИУ6 - 42Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**Г. Д. Нефедов

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(Подпись, дата) (И.О. Фамилия)

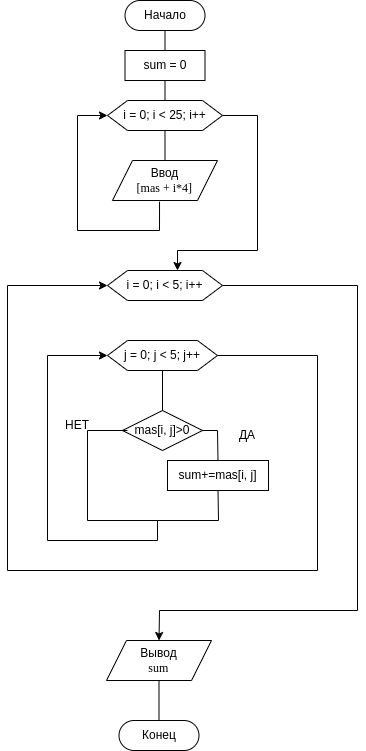
Москва, 2024

**Цель работы:** Изучение приемов моделирования обработки массивов и матриц в языке ассемблера.

**Задание**

Дана матрица 5х5. Определить сумму положительных элементов над побочной диагональю. Организовать ввод матрицы и вывод результатов.

Схема алгоритма показана на рисунке 1:

Рисунок 1 — Схема алгоритма

**Код программы:**

%include "lib.asm"

section .data

enterMsg db "Enter Array by lines:",10

lenEnterMsg equ $-enterMsg

outPut db "Your result: ", 10

lenOutMsg equ $-outPut

count dd 0

section .bss

mas resw 25

inbuf resw 10

lenIn equ $-inbuf

outbuf resb 10

lenout equ $-outbuf

result resd 1

section .text

global \_start

\_start:

; выводим сообщение о необходимости ввода

mov eax, 4 ; системная функция 4 (write)

mov ebx, 1 ; дескриптор файла stdout=1

mov ecx, enterMsg ; адрес выводимой строки

mov edx, lenEnterMsg ; длина выводимой строки

int 80h ; вызов системной функции

mov ECX, 0 ; счетчик для ввода чисел

input:

push ECX ; сохраняем значение счетчика цикла в стек

mov eax, 3 ; системная функция 3 (read)

mov ebx, 0 ; дескриптор файла stdin=0

mov ecx, inbuf ; адрес буфера ввода

mov edx, lenIn ; размер буфера

int 80h ; вызов системной функции

; передаем значние в функцию

mov esi,inbuf ; адрес введенной строки

call StrToInt

cmp EBX, 0 ; проверка кода ошибки

pop ECX ; возвращаем значение счетчика цикла из стека

mov [mas + ECX\*4], eax ; запись числа в память

inc ECX ; увеличиваем счетчик

cmp ECX, 25 ; ввели меньше 24 чисел? тогда продолжаем

jne input

mov edi, 4 ;записываем в регистр кол-во элементов в строке, до которой будем считать сумму

mov ecx, 4 ;записываем в регистр кол-во столбцов

mov edx, 0 ;здесь будем подсчитывать сумму элементов

mov ebx, 0 ;счетчик элементов

mov eax, 1 ;сюда будем записывать кол-во элементов через которые будем перескакивать

cycle:

push ecx ;закинули в стек ecx

mov ecx, edi ;записали в регистр ecx значение кол-ва элементов до которых будем считать

dec edi ;каждый раз уменьшаем количество элементов на 1

push eax ;закинем в стек значение региста eax

.cycle:

mov eax, [mas + ebx\*4] ;запишем в eax значение текущего элемента

inc ebx ;прибавляем 1 для подсчета следующего элемента

cmp eax, 0 ;сравниваем текущее значение с 0

jl .cycle ;если меньше 0, то переходим к след элементу без добавления текущего элемента к сумме

add edx, eax ;добавляем значение текущего элемента в общуюю сумму

loop .cycle ;возвращаемся на метку по суммированию строк

pop eax ;возвращаем кол-во элементов через которые будем перескакивать

add ebx, eax ;суммируем

inc eax ;инкрементируем кол-во элементов через которые будем перескакивать

pop ecx ;возвращаем кол-во столбцов

loop cycle ;возвращаемся к циклу

jmp print

print:

mov [result], edx ;переписываем значение регистра edx в переменную

mov eax, [result]

mov esi, outbuf

cwde

call IntToStr

;; выводим результат на экран

mov eax, 4

mov ebx, 1

mov ecx, outPut

mov edx, lenOutMsg

int 0x80

mov eax, 4

mov ebx, 1

mov ecx, outbuf

mov edx, eax

int 0x80

mov eax, 1

xor ebx, ebx

int 0x80

Отладка программы продемонстрирована на рисунках 2 -

Введем в программу следующую матрицу:

1,2,-3,4,5

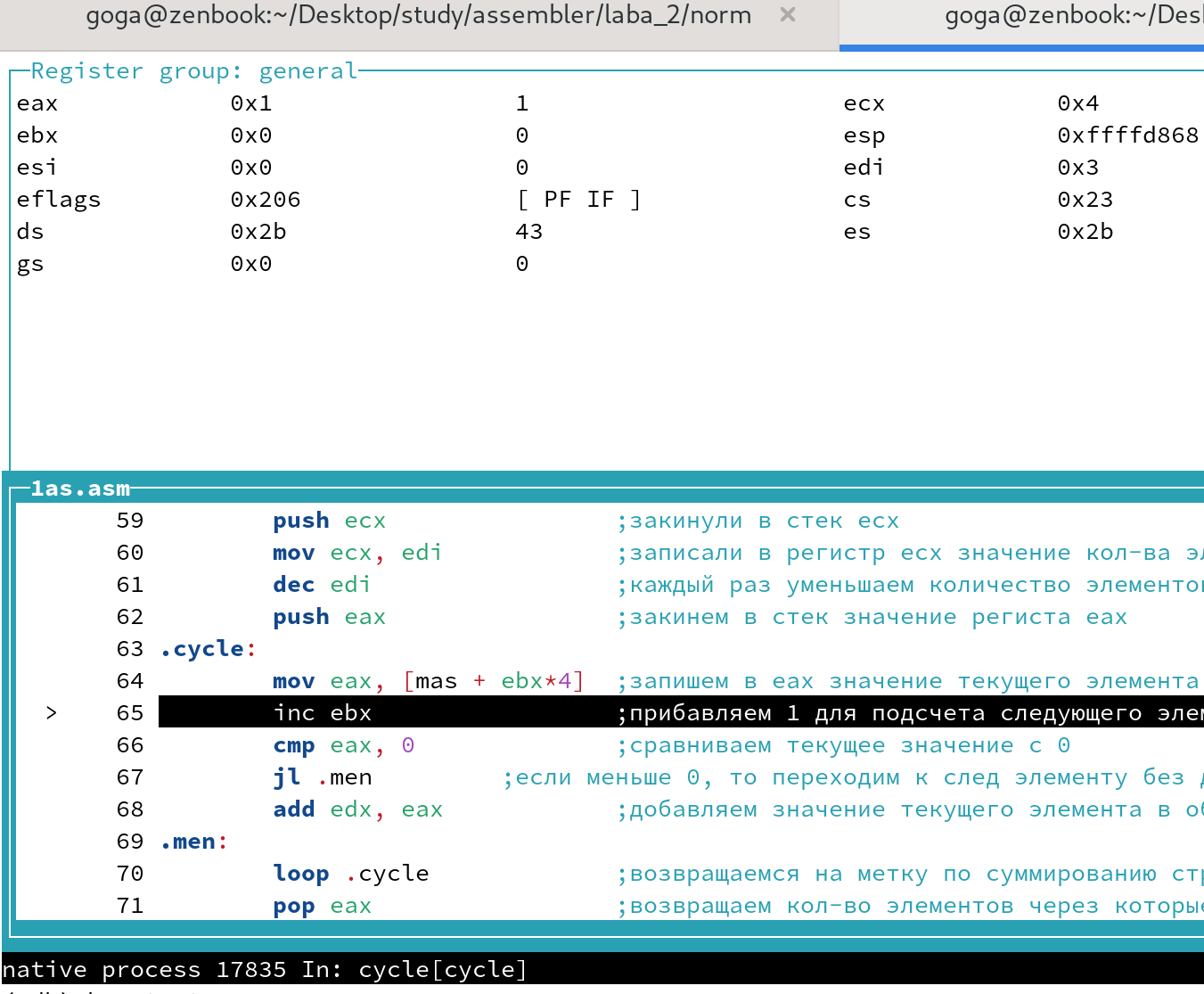
5,6,7,8,9

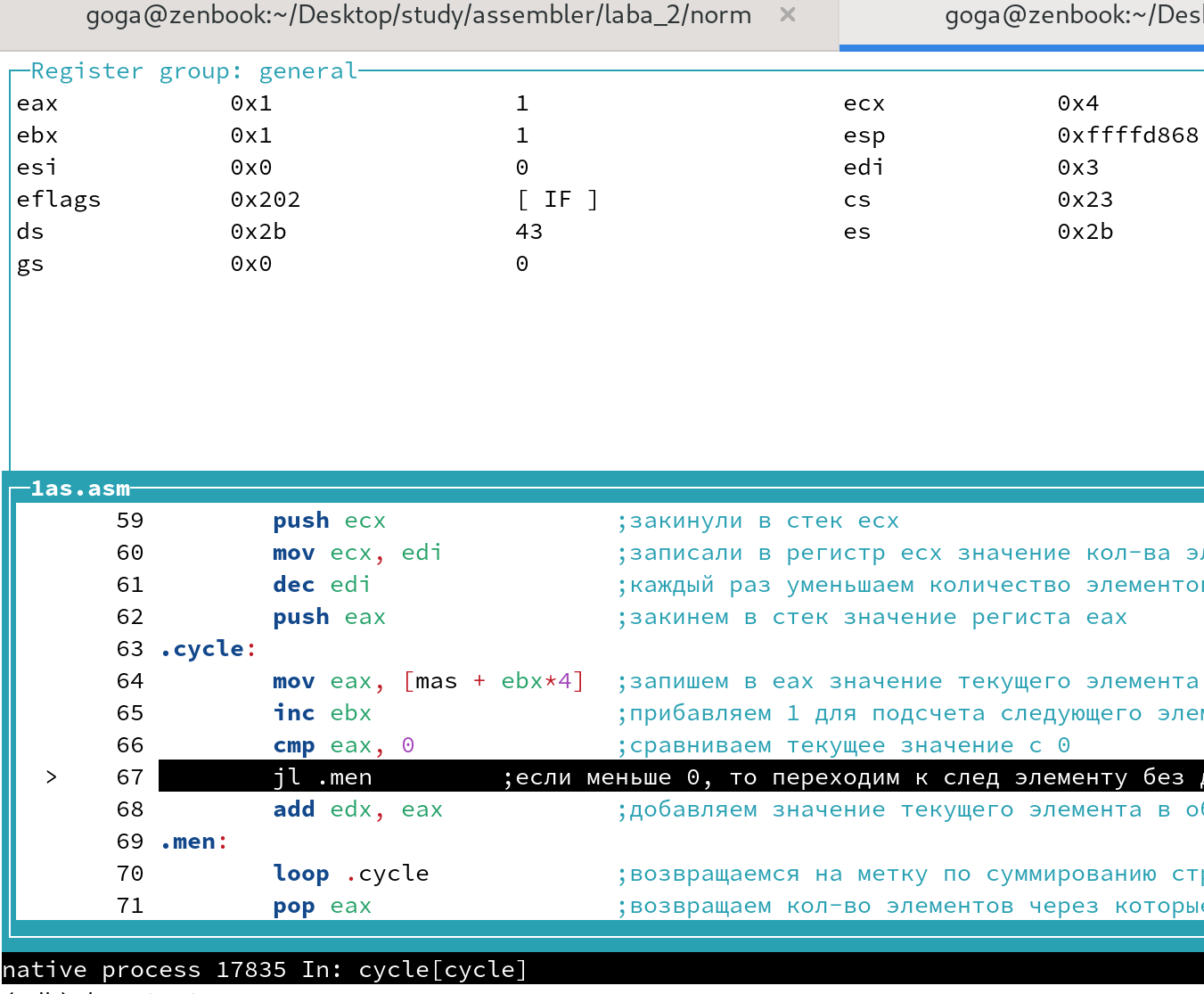
5,-6,7,8,9

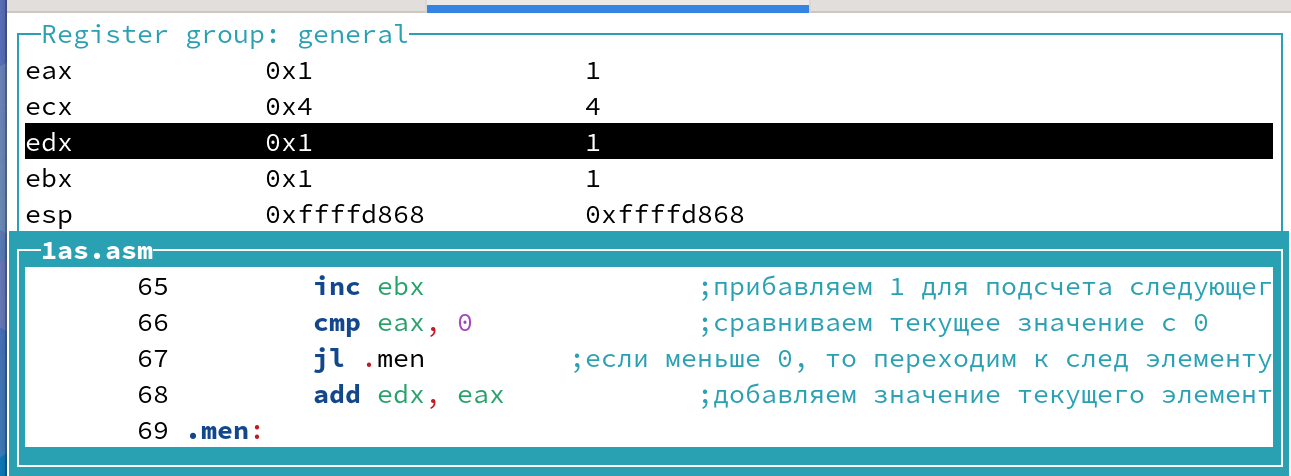
5,6,7,8,9

5,6,7,8,9

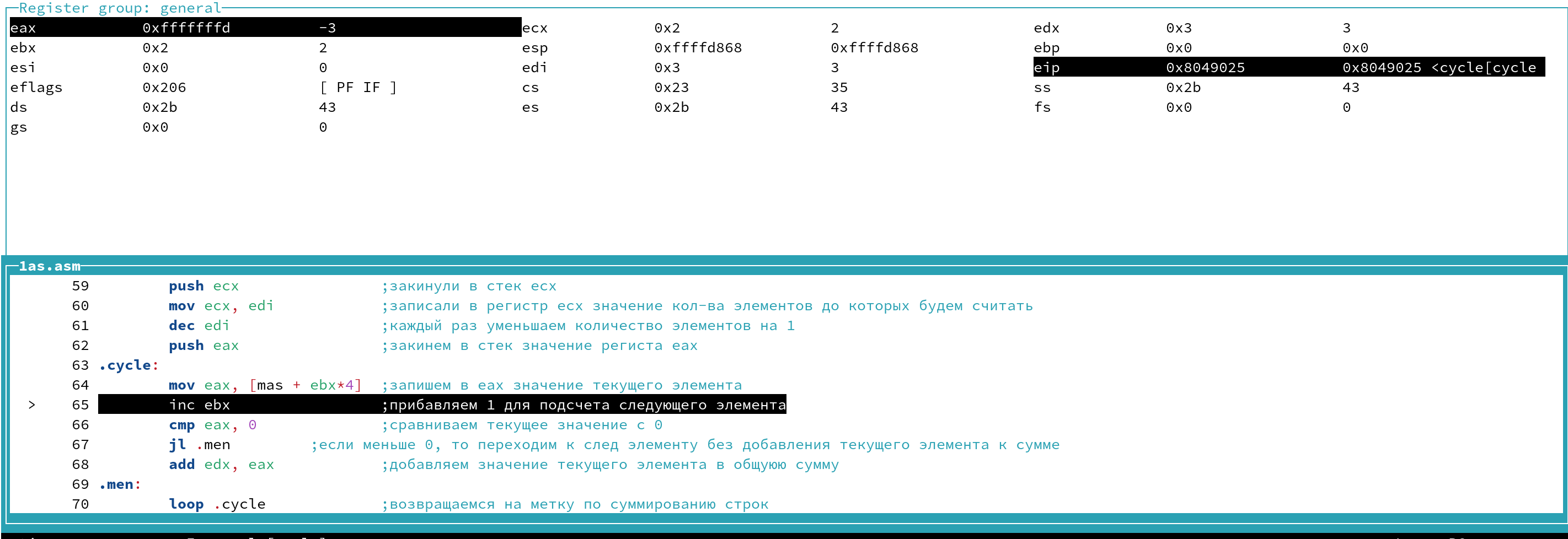
Проверим отладку на положительном элементе матрицы, рисунки 2 - 4:

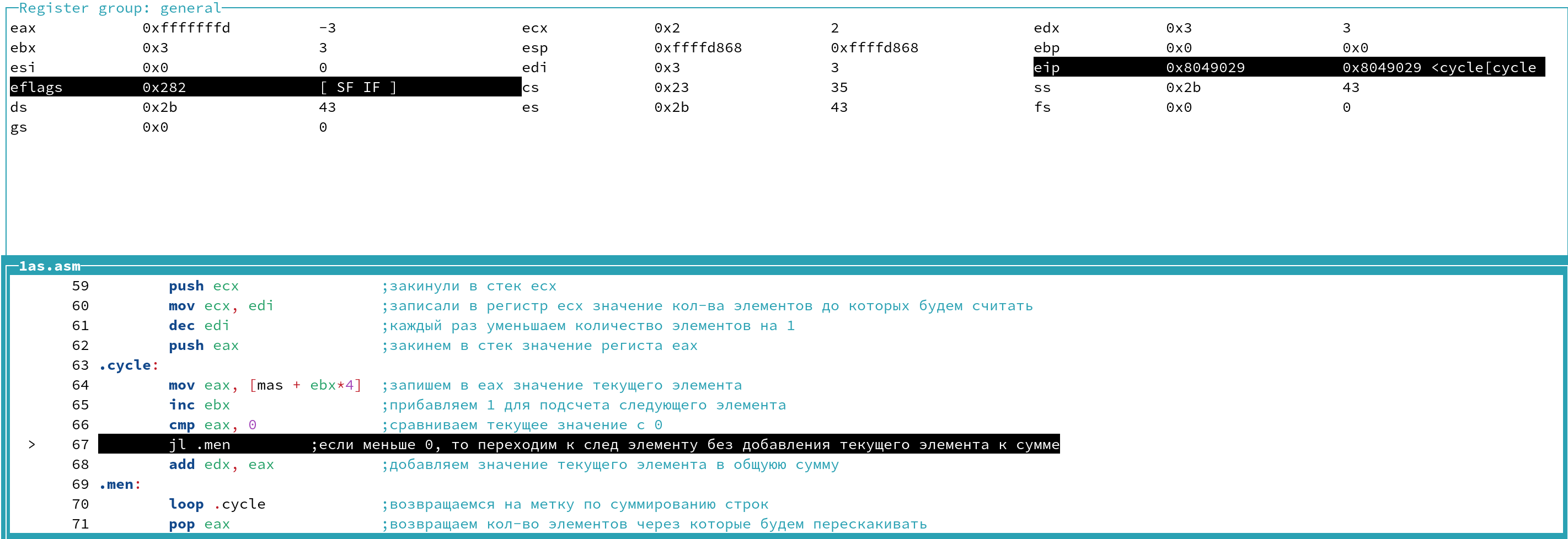
Рисунок 2 — Запись элемента матрицы в регистр eax

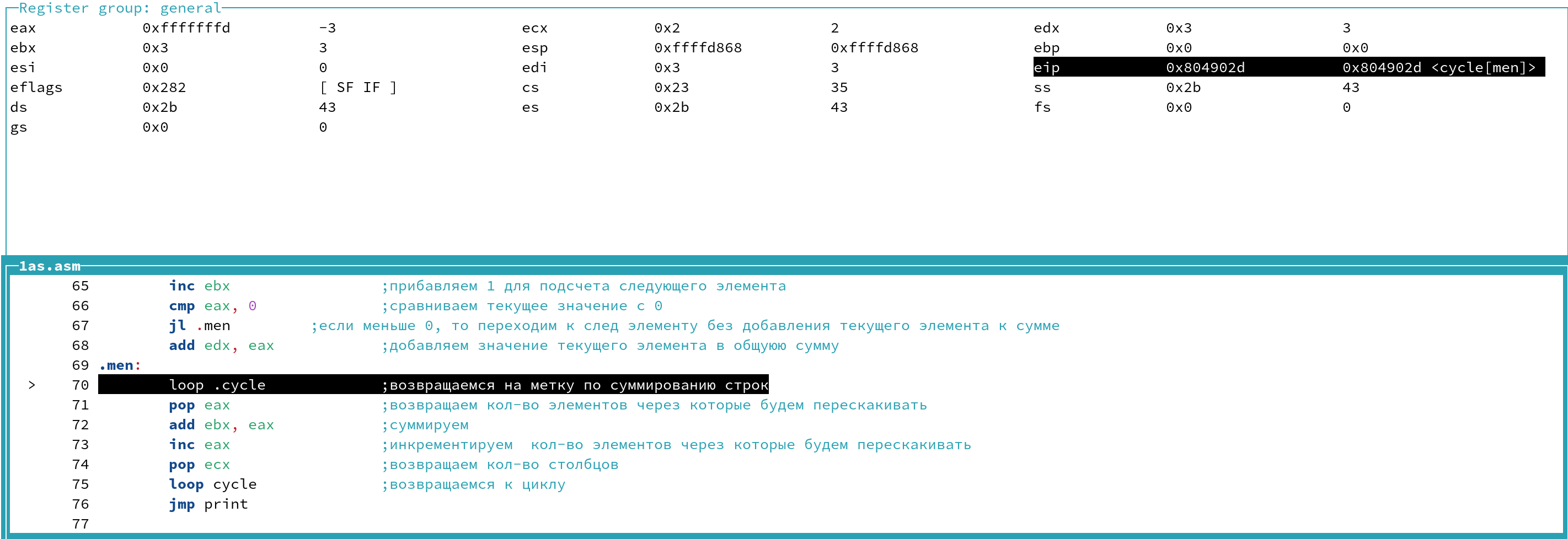
Рисунок 3 — Сравнили элемент с 0

Рисунок 4 — Прибавили значение элемента массива к общей сумме

Теперь проверим отладку на положительном элементе матрицы, рисунки 5 - 7:

Рисунок 5 — Получение отрицательного элемента «-3»

Рисунок 6 — Сравнение eax c 0, результат этой операции — флаг Sign Flag

Рисунок 7 — Переход по метки .men

Проведем тестирование программы, результаты приведены в таблице 1:

|  |  |  |
| --- | --- | --- |
| Исходные данные | Ожидаемый результат | Полученный результат |
| mas dd 1,2,-3,4,5  dd 5,6,7,8,9  dd 5,-6,7,8,9  dd 5,6,7,8,9  dd 5,6,7,8,9 | 35 | 35 |
| mas dd -1,-2,-3,-4,5  dd -5,-6,-7,8,9  dd 0,-6,0,8,9  dd 0,0,7,8,9  dd 5,6,7,8,9 | 0 | 0 |
| mas dd -1,-2,-3,-4,5  dd -5,-6,-7,8,9  dd 0,6,0,8,9  dd 0,0,7,8,9  dd 5,6,7,8,9 | 6 | 6 |
| mas dd -1,-2,-3,-4,5  dd -115,-16,-7,8,9  dd 10,116,0,8,9  dd 100,-100,7,8,9  dd 5,6,7,8,9 | 226 | 226 |

**Контрольные вопросы:**

**1. Почему в  ассемблере не  определены понятия  «массив»,  «матрица»?**

В ассемблере отсутствие явного определения "массивов" и "матриц" связано с тем, что память имеет плоскую модель. В такой модели памяти все данные хранятся последовательно в одном адресном пространстве, и для доступа к этим данным используются адреса, начиная с нулевого адреса.

**2. Как  в  ассемблере  моделируются  массивы?**

array db 10, 20, 30, 40, 50 ; массив байтов

**3.Поясните фрагмент последовательной адресации элементов массива? Почему при этом для хранения частей адреса используют регистры?**

При последовательной адресации элементов массива используются регистры для хранения адреса элемента и его последующего увеличения на один после каждой итерации цикла. Это позволяет обращаться к элементам массива последовательно, обеспечивая эффективное выполнение цикла.

**4. Как в памяти компьютера размещаются элементы матриц?**

В памяти компьютера размещаются элементы матриц последовательно.

**5.Чем моделирование матриц отличается от моделирования массивов? В каких случаях при выполнении операций для адресации матриц используется один регистр, а в каких – два?**  
При работе с матрицами используются два вложенных цикла: один для перебора строк матрицы, а другой - для перебора столбцов в каждой строке. Это позволяет обращаться к каждому элементу матрицы поочередно, по всем строкам и столбцам.