Отчет по лабораторной работе №7

Дисциплина: архитектура компьютера

Бородин Дмитрий Алексеевич

Содержание

1 Цель работы

Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Задание

1. Реализация переходов в NASM
2. Изучение структуры файлов листинга
3. Самостоятельное написание программ по материалам лабораторной работы

3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку про- граммы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы №7 (рис. 1).

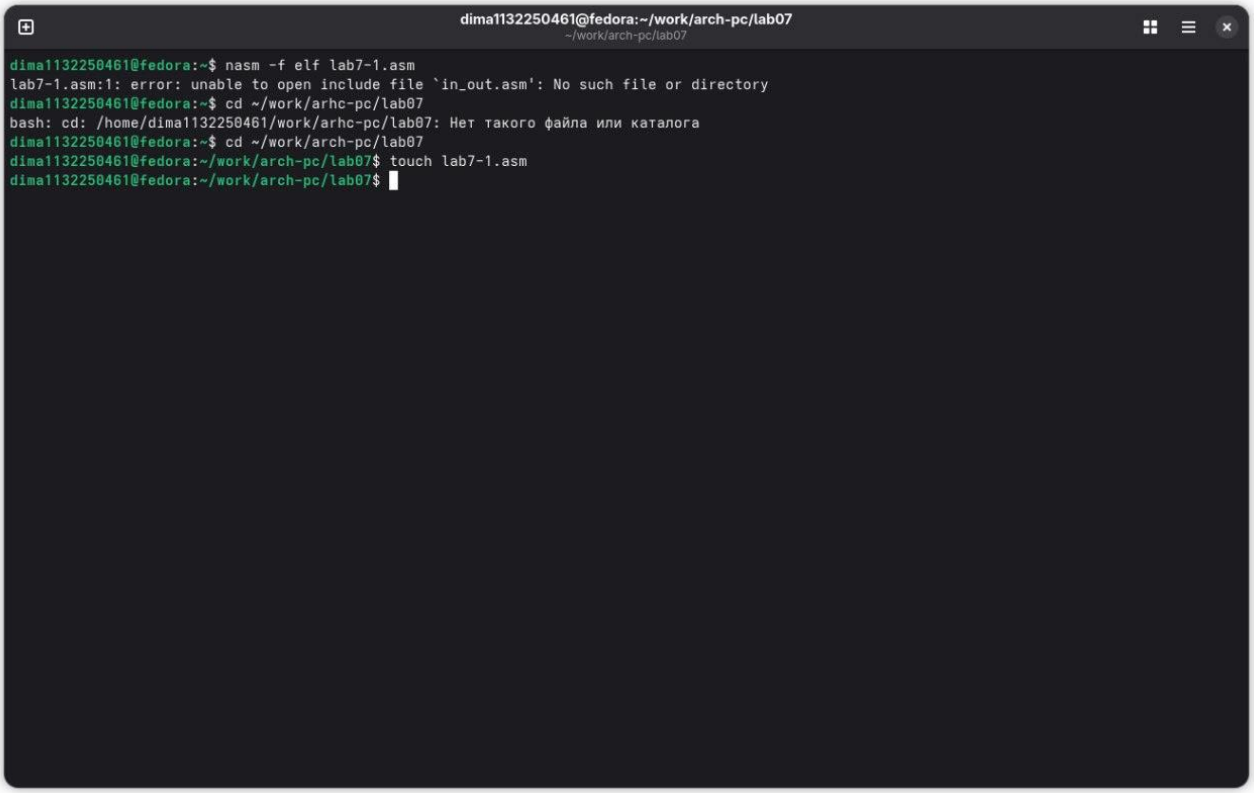

Рис. 1: Создание каталога и файла для программы

Копирую код из листинга в файл будущей программы. (рис. 2).

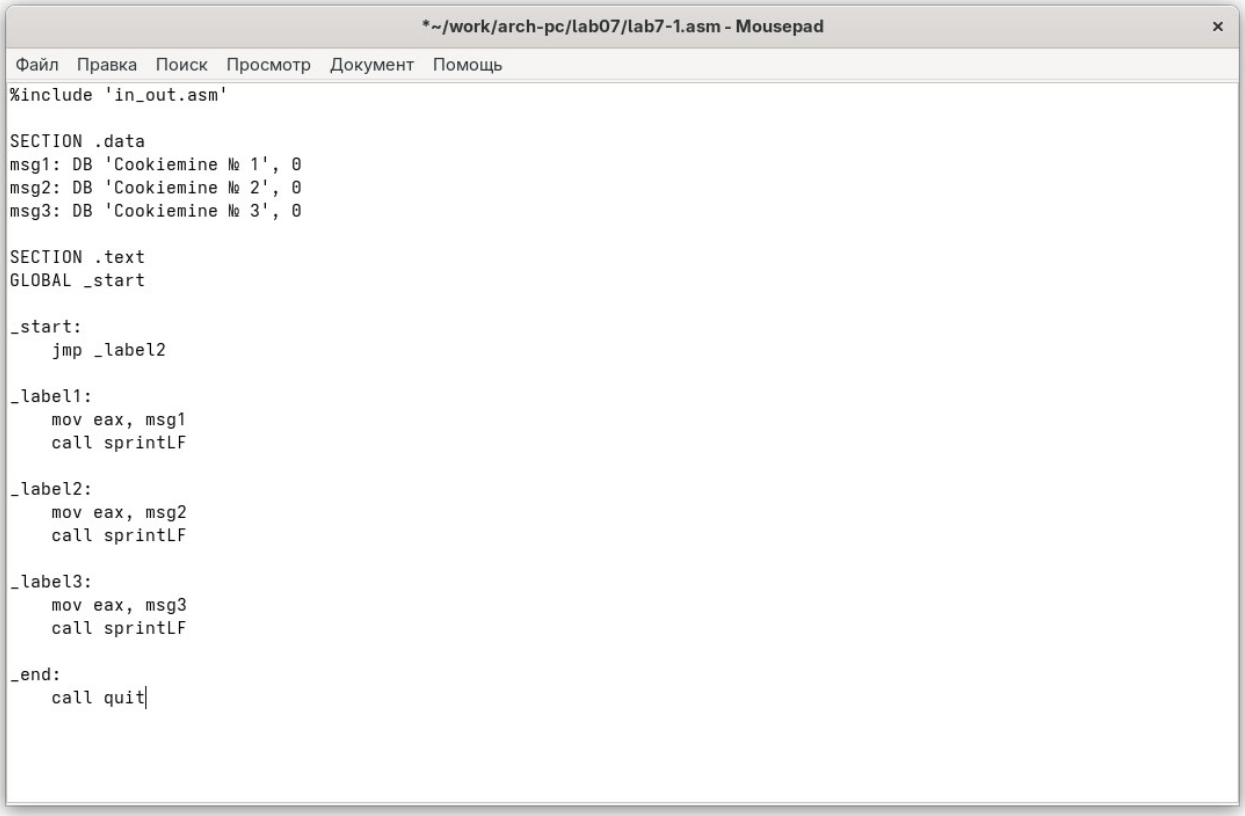

Рис. 2: Сохранение программы

При запуске программы я убедился в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

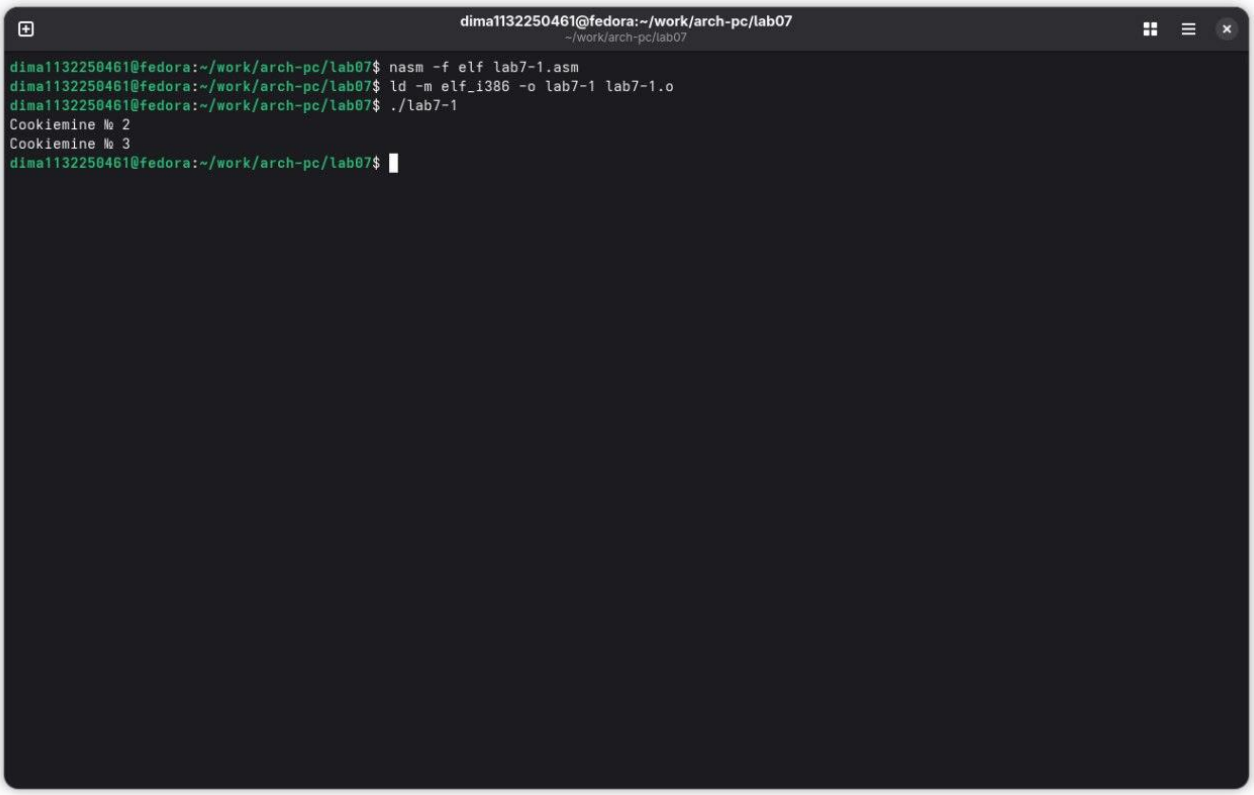

Рис. 3: Запуск программы

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций (рис. 4).

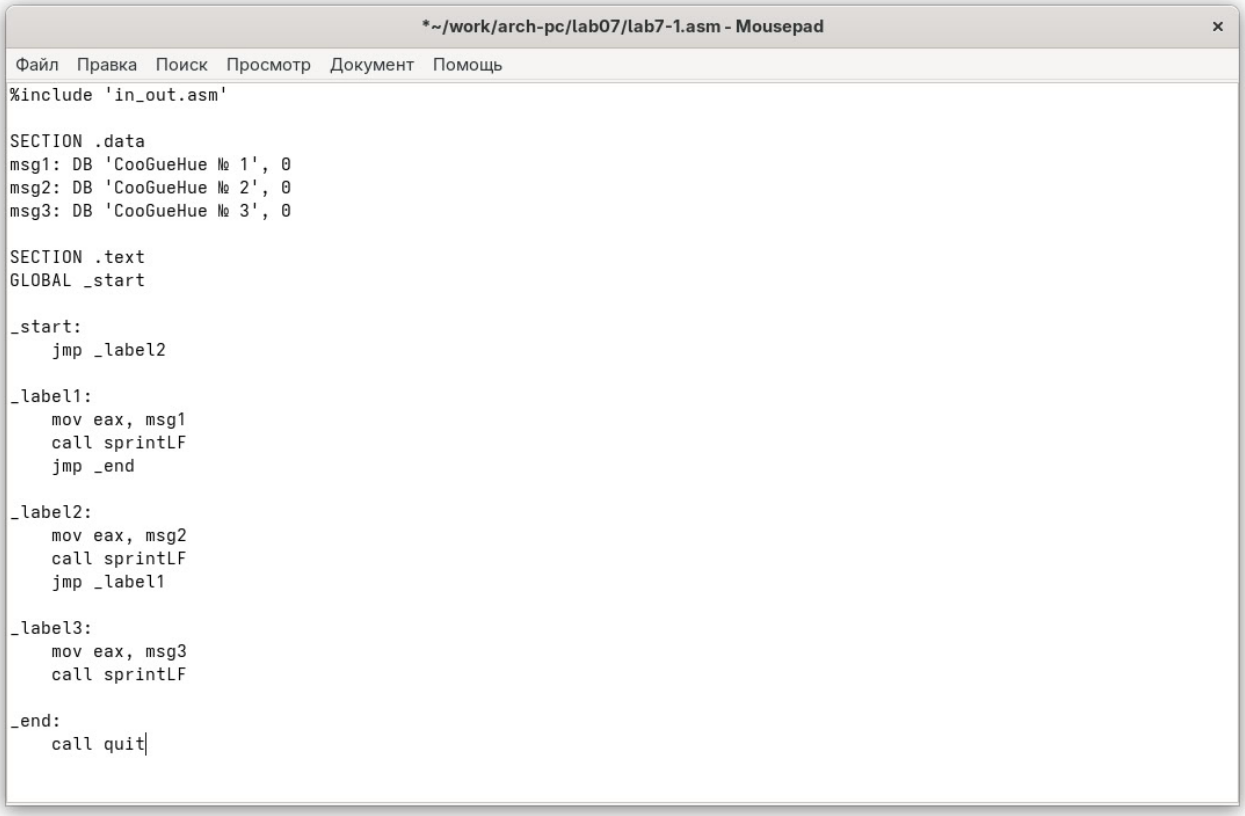

Рис. 4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны (рис. 5).

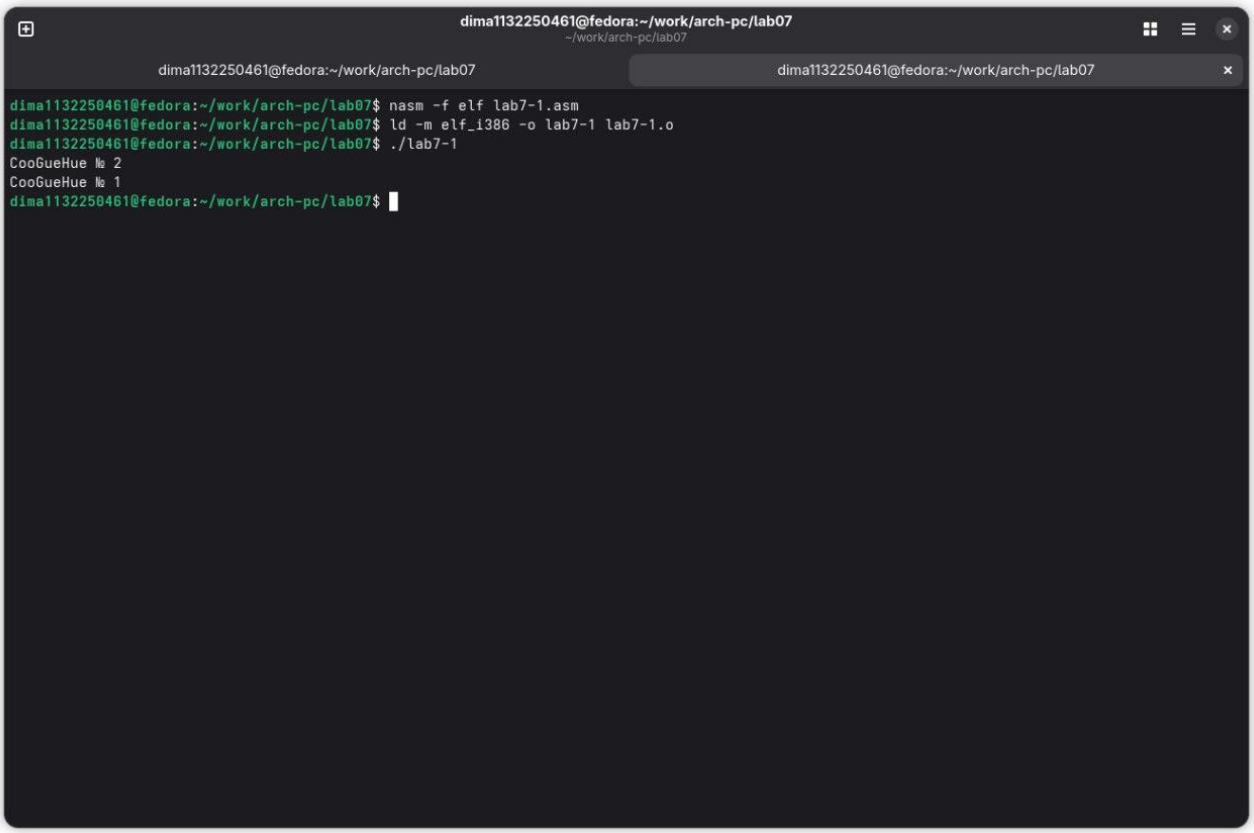

Рис. 5: Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке (рис. 6).

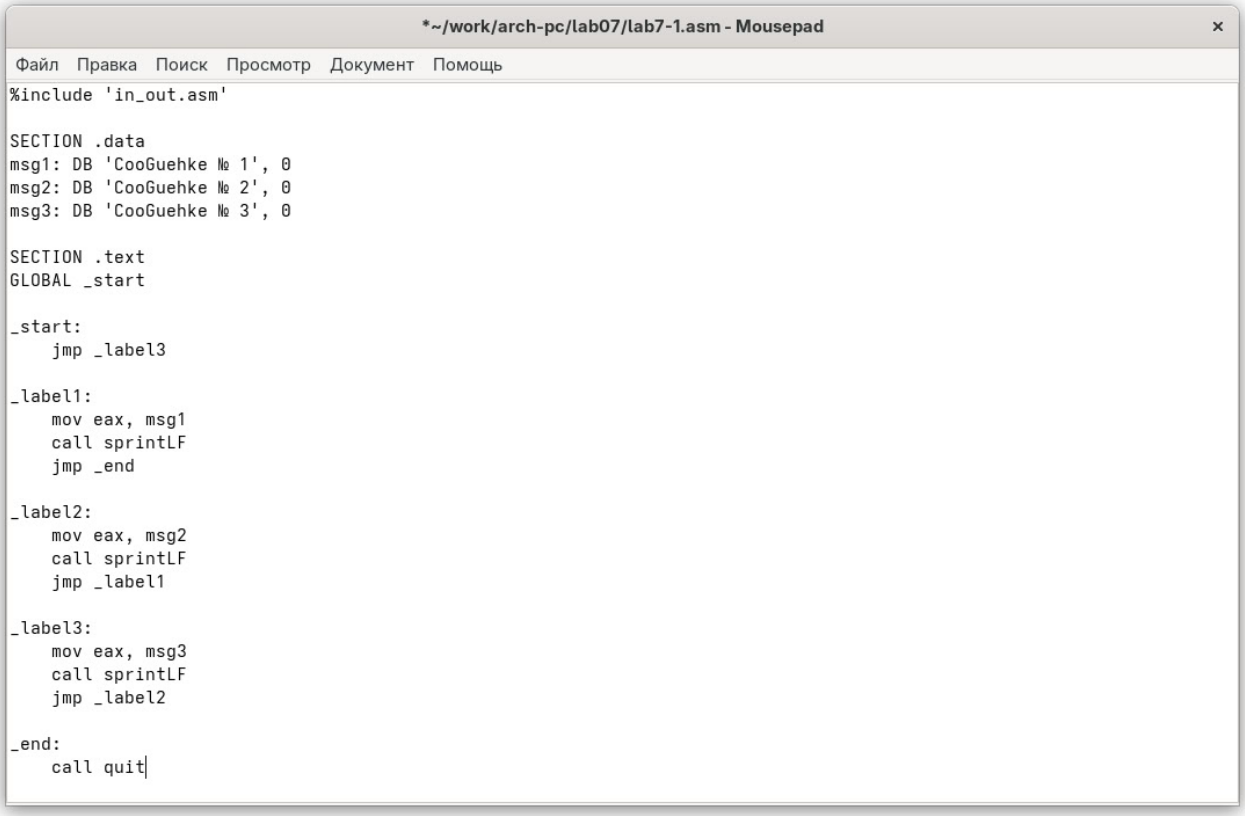

Рис. 6: Изменение программы

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).

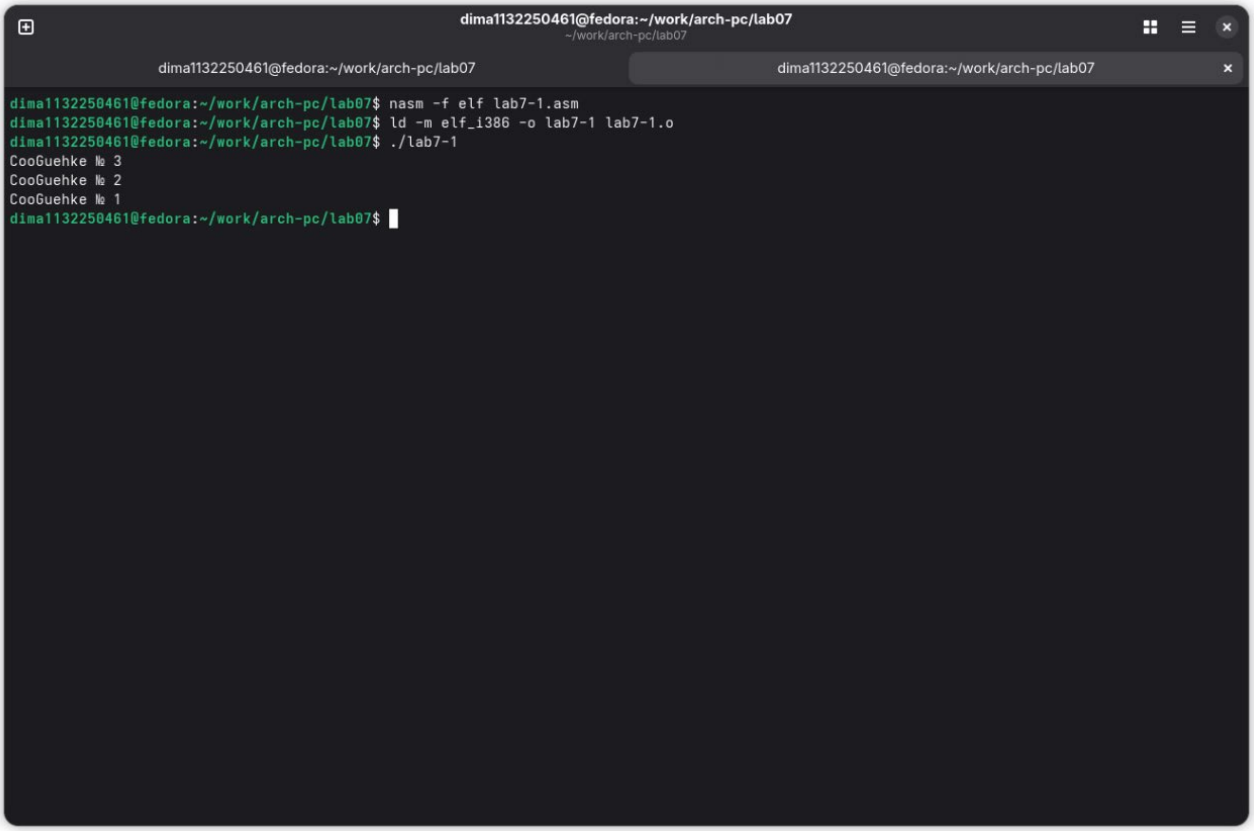

Рис. 7: Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).

*~/work/arch-pc/lab07/lab7-2.asm - Mousepad

Файл Правка Поиск Просмотр Документ Помощь

```
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наибольшее число: ', 0h
A dd 20
C dd 50

SECTION .bss
max resb 10
B resb 10

SECTION .text
GLOBAL _start

_start:
    mov eax, msg1
    call sprint

    mov ecx, B
    mov edx, 10
    call sread

    mov eax, B
    call atoi
    mov [B], eax

    mov ecx, [A]
    mov [max], ecx

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
    mov [max], ecx

check_B:
    mov eax, [max]
    mov ecx, eax

    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx

fin:
    mov eax, msg2
    call sprint
    mov eax, [max]
    call iprintLF
    call quit
```

Рис. 8: Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

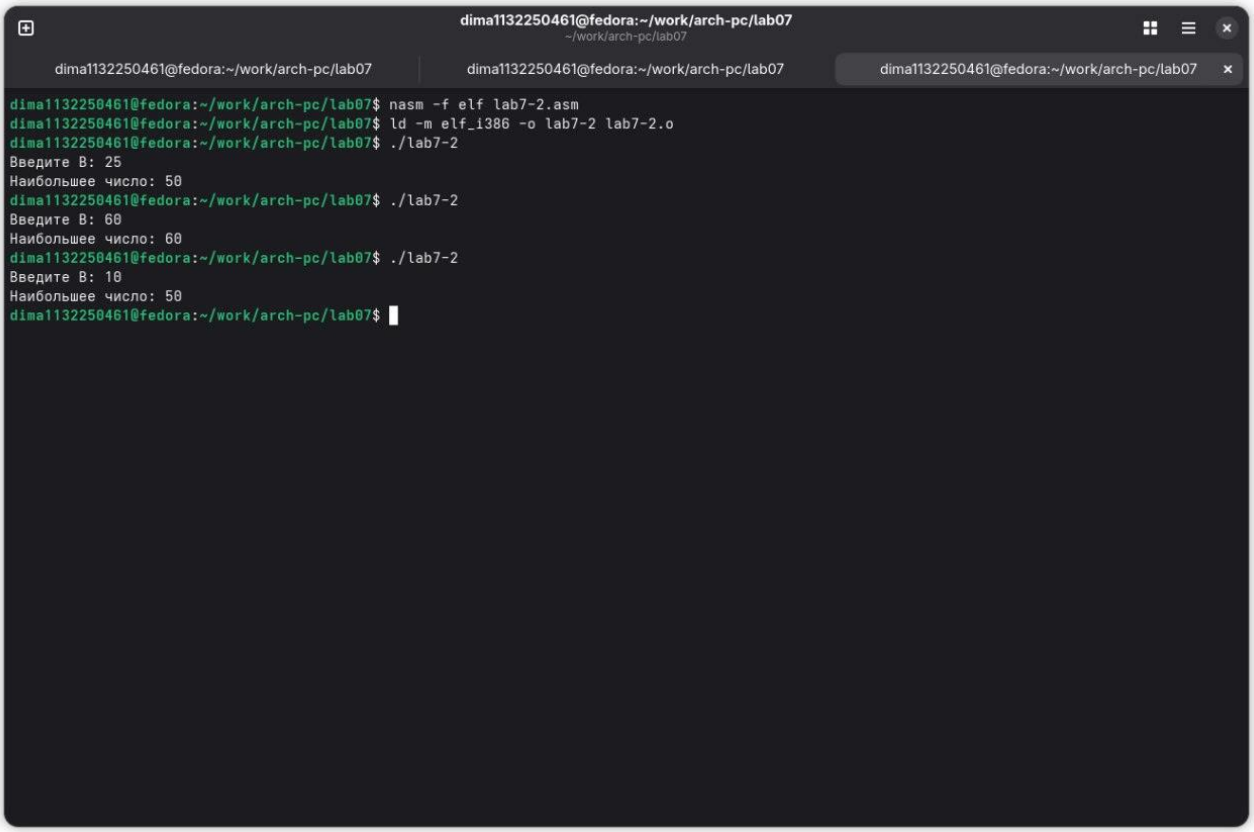

Рис. 9: Проверка программы из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mousepad (рис. 10).

~/work/arch-pc/lab07/lab7-2.lst - Mousepad

Файл Правка Поиск Просмотр Документ Помощь

```
    1                                   %include 'in_out.asm'
    1                                <1> ;---------------   slen  ------------
    2                                <1> ; Функция вычисления длины сообщения
    3                                <1> slen:
    4 00000000 53                    <1>     push    ebx
    5 00000001 89C3                  <1>     mov     ebx, eax
    6                                <1>
    7                                <1> nextchar:
    8 00000003 803800                <1>     cmp     byte [eax], 0
    9 00000006 7403                  <1>     jz      finished
   10 00000008 40                    <1>     inc     eax
   11 00000009 EBF8                  <1>     jmp     nextchar
   12                                <1>
   13                                <1> finished:
   14 0000000B 29D8                  <1>     sub     eax, ebx
   15 0000000D 5B                    <1>     pop     ebx
   16 0000000E C3                    <1>     ret
   17                                <1>
   18                                <1>
   19                                <1> ;---------------  sprint  -----------
   20                                <1> ; Функция печати сообщения
   21                                <1> ; входные данные: mov eax,<message>
   22                                <1> sprint:
   23 0000000F 52                    <1>     push    edx
   24 00000010 51                    <1>     push    ecx
   25 00000011 53                    <1>     push    ebx
   26 00000012 50                    <1>     push    eax
   27 00000013 E8E8FFFFFF            <1>     call    slen
   28                                <1>
   29 00000018 89C2                  <1>     mov     edx, eax
   30 0000001A 58                    <1>     pop     eax
   31                                <1>
   32 0000001B 89C1                  <1>     mov     ecx, eax
   33 0000001D BB01000000            <1>     mov     ebx, 1
   34 00000022 B804000000            <1>     mov     eax, 4
   35 00000027 CD80                  <1>     int     80h
   36                                <1>
   37 00000029 5B                    <1>     pop     ebx
   38 0000002A 59                    <1>     pop     ecx
   39 0000002B 5A                    <1>     pop     edx
   40 0000002C C3                    <1>     ret
   41                                <1>
   42                                <1>
   43                                <1> ;---------------  sprintLF  --------
   44                                <1> ; Функция печати сообщения с переводс
   45                                <1> ; входные данные: mov eax,<message>
   46                                <1> sprintLF:
   47 0000002D E8DDFFFFFF            <1>     call    sprint
```

Рис. 10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст прогарммы с комментариями.

Удаляю один операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

*~/work/arch-pc/lab07/lab7-2.asm - Mousepad

Файл Правка Поиск Просмотр Документ Помощь

```
SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наибольшее число: ', 0h
A dd 20
C dd 50

SECTION .bss
max resb 10
B resb 10

SECTION .text
GLOBAL _start

_start:
    mov eax, msg1
    call sprint

    mov ecx, B
    mov edx, 10
    call sread

    mov eax, B
    call atoi
    mov [B], eax

    mov ecx, [A]
    mov [max], ecx

    cmp ecx, [C]
    jg check_B
    mov ecx, [C]
    mov [max], ecx

check_B:
    mov eax,
    mov ecx, eax

    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx

fin:
    mov eax, msg2
    call sprint
    mov eax, [max]
    call iprintLF
    call quit
```

Рис. 11: Удаление операнда из программы

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 12).

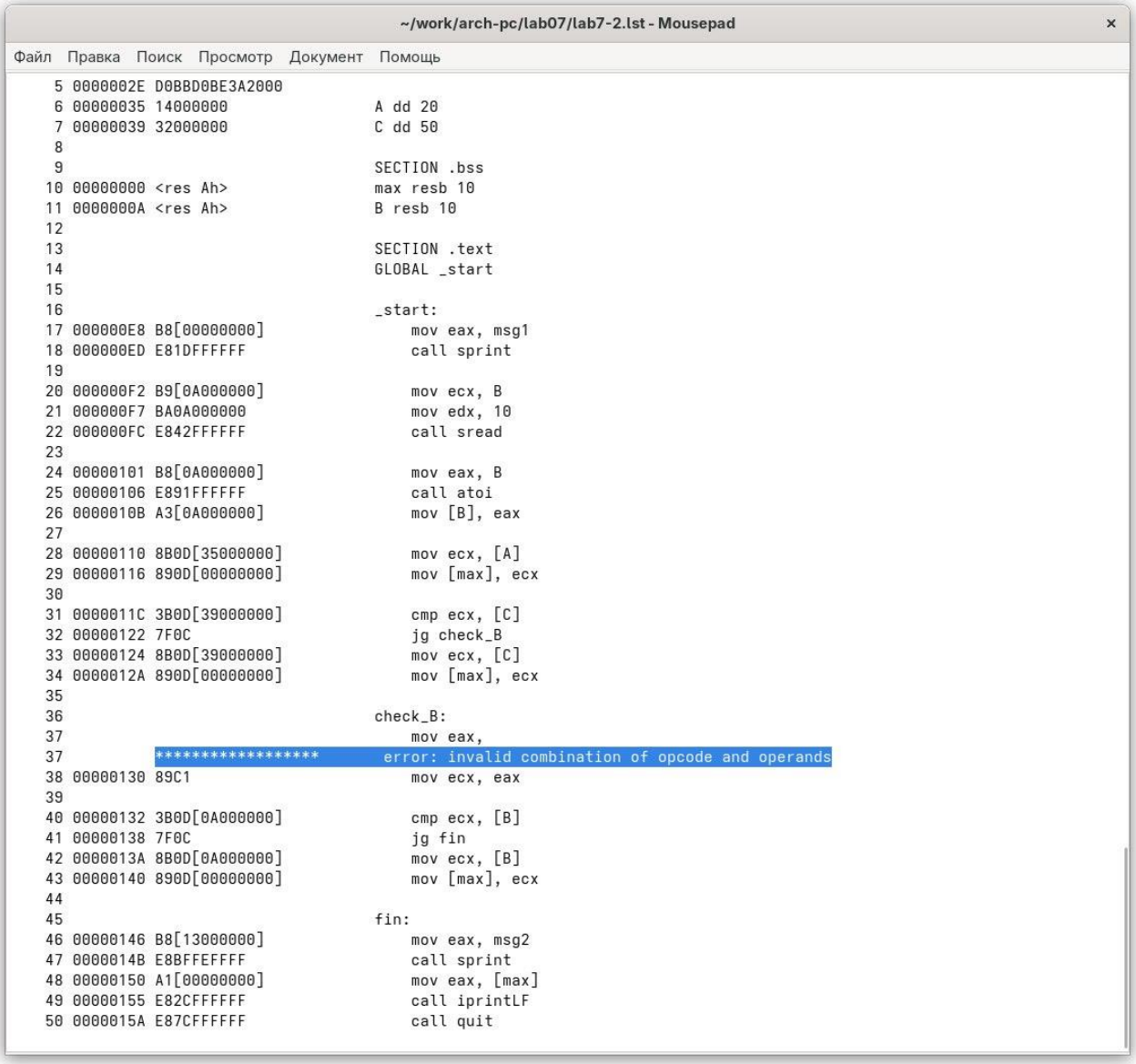

Рис. 12: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

Искренне не понимаю, какой вариант я должен был получить во время 7 лабораторной работы, поэтому буду использовать свой вариант - девятый - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением (рис. 13).

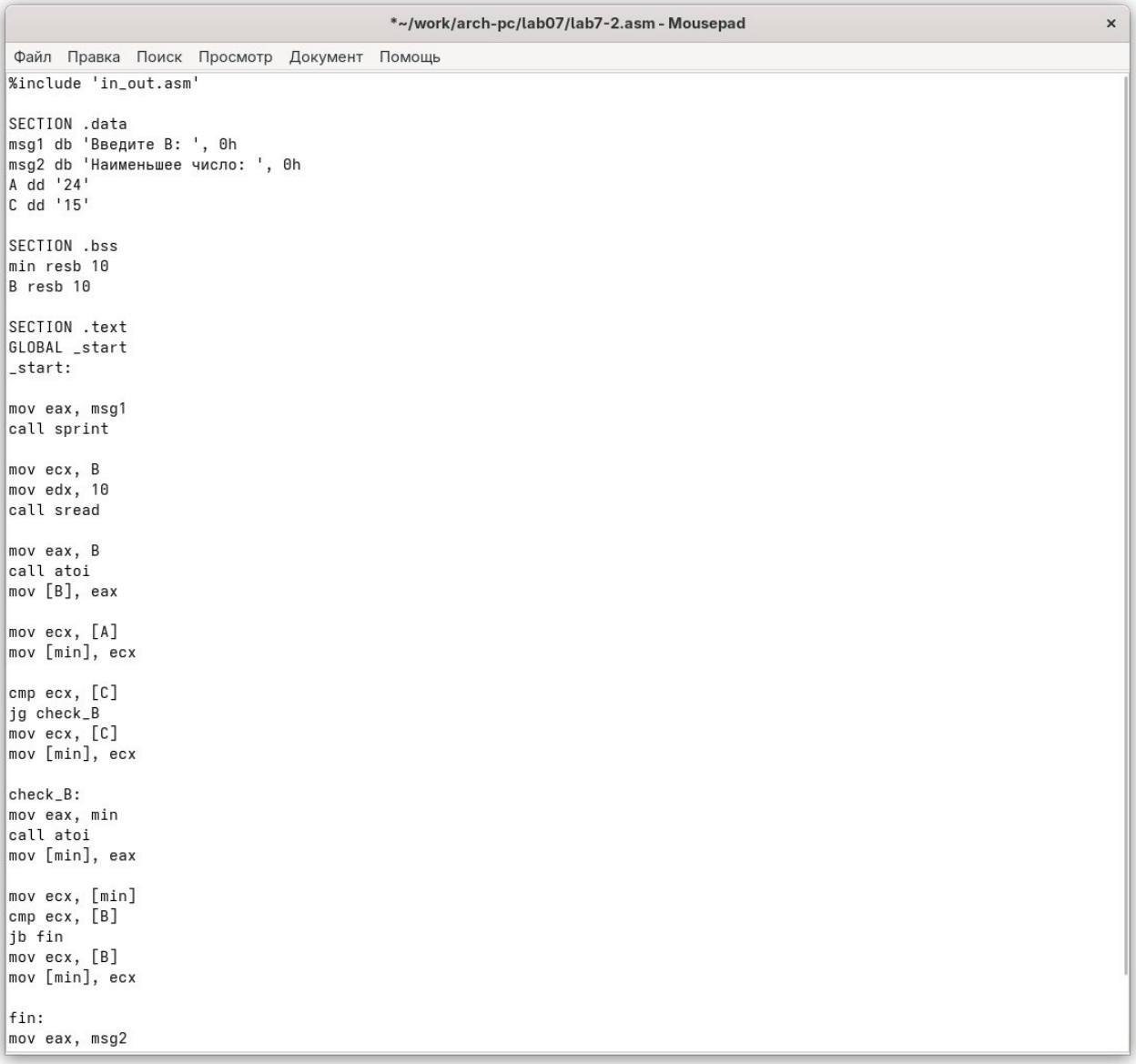

Рис. 13: Первая программа самостоятельной работы

Код первой программы:

```
%include 'in_out.asm'

SECTION .data
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'

SECTION .bss
min resb 10
B resb 10
```

```
SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint

mov ecx, B
mov edx, 10
call sread

mov eax, B
call atoi
mov [B], eax

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit
```

Проверяю корректность написания первой программы (рис. 14).

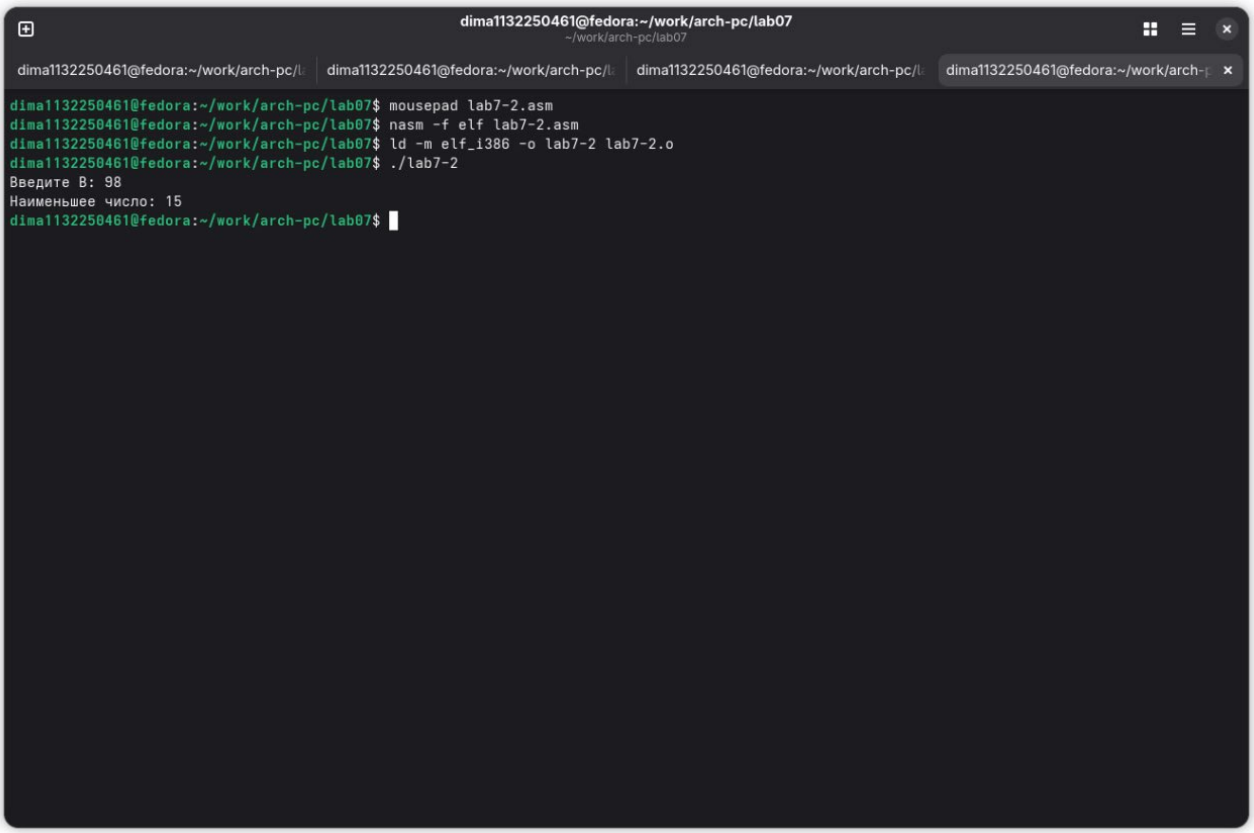

Рис. 14: Проверка работы первой программы

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатурых переменных a и x (рис. 15).

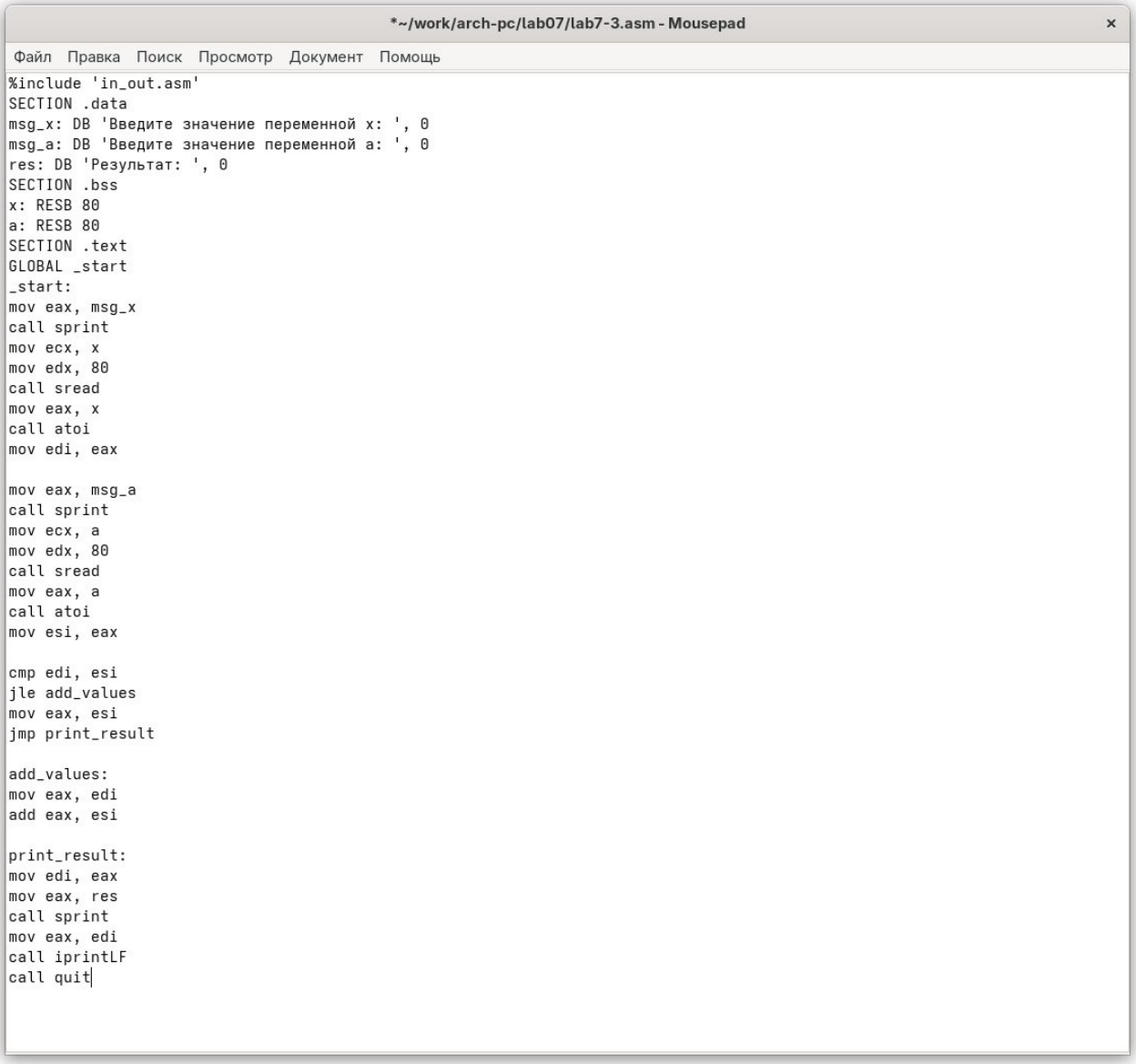

Рис. 15: Вторая программа самостоятельной работы

Код второй программы:

```
%include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0
SECTION .bss
x: RESB 80
a: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
```

```
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp edi, esi
jle add_values
mov eax, esi
jmp print_result

add_values:
mov eax, edi
add eax, esi

print_result:
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit
```

Транслирую и компоную файл, запускаю и проверяю работу программмы для различных значений a и x (рис. 16).

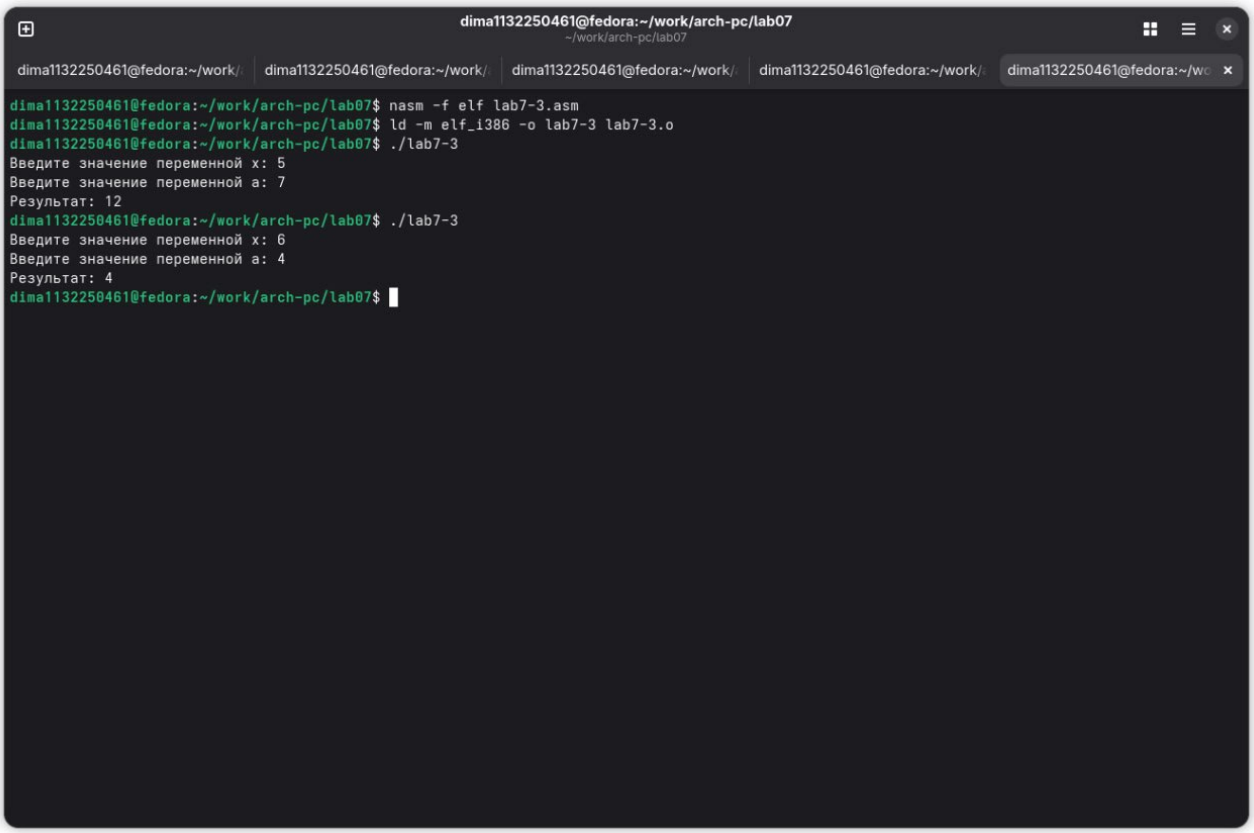

Рис. 16: Проверка работы второй программы

5 Выводы

При выполнении лабораторной работы я изучил команды условных и безусловных переходво, а также приобрел навыки написания программ с использованием перходов, познакомился с назначением и структурой файлов листинга.

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №7
3. Программирование на языке ассемблера NASM Столяров А. В.