

# **Отчёт по лабораторной работе №8**

**дисциплина: Архитектура компьютера**

**Сычев Егор Олегович**

# Содержание

|   |                                |    |
|---|--------------------------------|----|
| 1 | Цель работы                    | 5  |
| 2 | Выполнение лабораторной работы | 6  |
| 3 | Самостоятельная работа         | 15 |
| 4 | Вывод                          | 20 |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 2.1  | Команды: mkdir, cd, touch . . . . .                 | 6  |
| 2.2  | mcedit . . . . .                                    | 7  |
| 2.3  | Транслирование, компоновка, запуск . . . . .        | 7  |
| 2.4  | mcedit . . . . .                                    | 8  |
| 2.5  | Транслирование, компоновка, запуск . . . . .        | 8  |
| 2.6  | mcedit . . . . .                                    | 9  |
| 2.7  | Транслирование, компоновка, запуск . . . . .        | 9  |
| 2.8  | Команды: touch, mcedit . . . . .                    | 9  |
| 2.9  | mcedit . . . . .                                    | 10 |
| 2.10 | Транслирование, компоновка, запуск . . . . .        | 10 |
| 2.11 | Транслирование с файлом листинга   mcedit . . . . . | 11 |
| 2.12 | Файл lab8-2.lst . . . . .                           | 11 |
| 2.13 | Строки 46-49 . . . . .                              | 12 |
| 2.14 | mcedit . . . . .                                    | 13 |
| 2.15 | Транслирование с файлом листинга   Ошибка . . . . . | 13 |
| 2.16 | Команда ls . . . . .                                | 13 |
| 2.17 | Файл lab8-2.lst . . . . .                           | 14 |
| 3.1  | Команды: touch, mcedit . . . . .                    | 15 |
| 3.2  | mcedit . . . . .                                    | 16 |
| 3.3  | Транслирование, компоновка, запуск . . . . .        | 16 |
| 3.4  | Функция и значения х,а для проверки . . . . .       | 17 |
| 3.5  | Команды: touch, mcedit . . . . .                    | 17 |
| 3.6  | mcedit . . . . .                                    | 18 |
| 3.7  | Транслирование, компоновка, запуск . . . . .        | 19 |

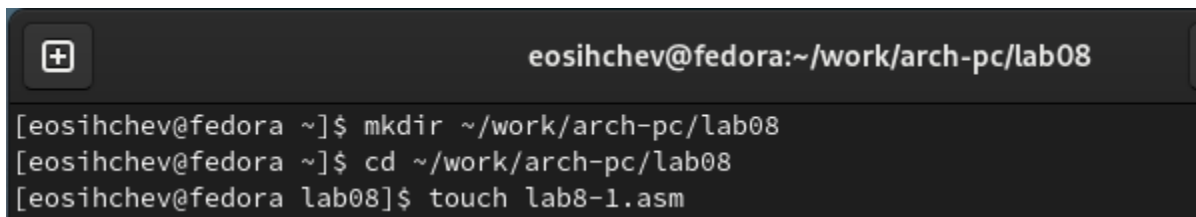
## Список таблиц

# 1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыки написания программ с использованием переходов. Ознакомиться с назначением и структурой файла листинга.

## 2 Выполнение лабораторной работы

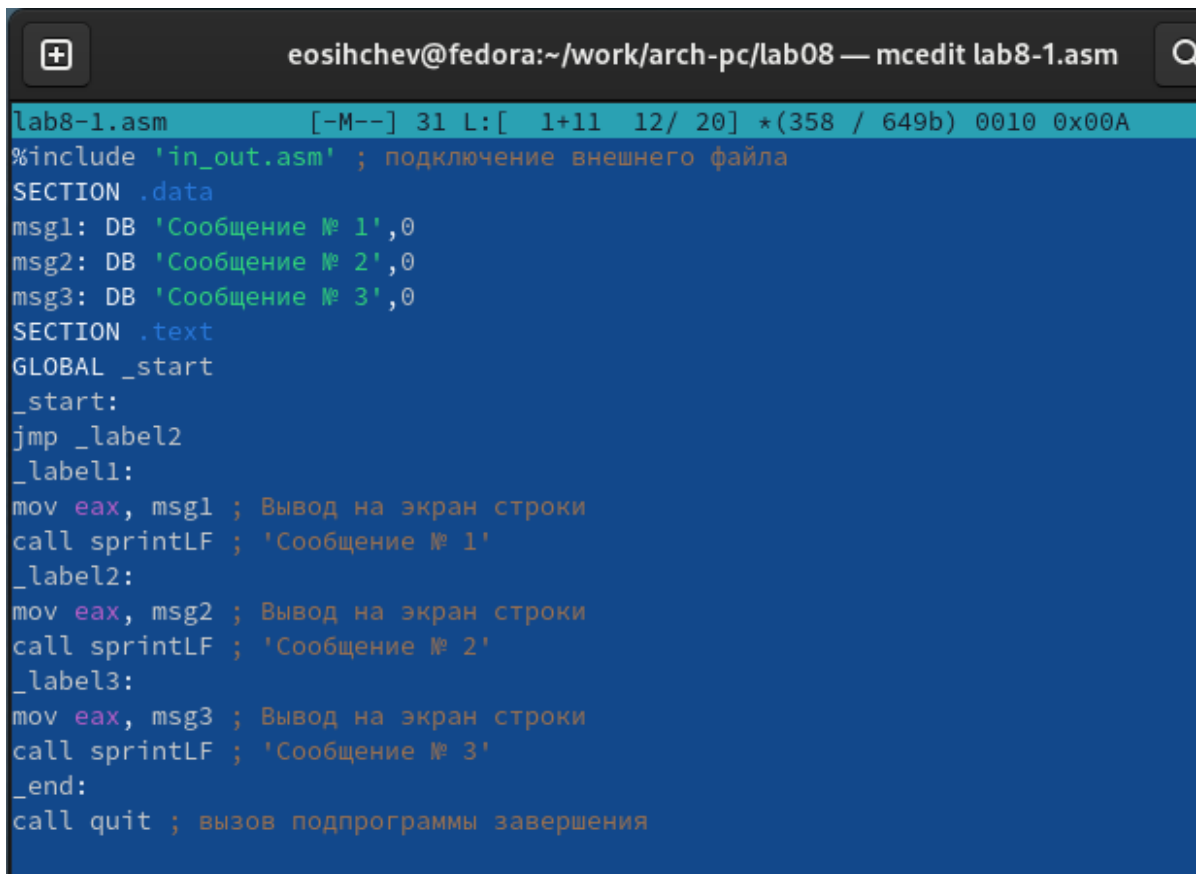
1. Создаем каталог для программ лабораторной работы №8 и в нем же создаем файл lab8-1.asm.



```
eosihchev@fedora:~/work/arch-pc/lab08  
[eosihchev@fedora ~]$ mkdir ~/work/arch-pc/lab08  
[eosihchev@fedora ~]$ cd ~/work/arch-pc/lab08  
[eosihchev@fedora lab08]$ touch lab8-1.asm
```

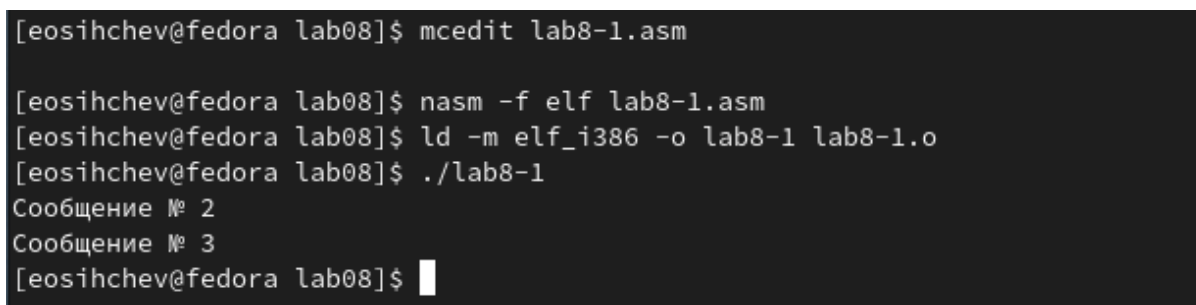
Рис. 2.1: Команды: mkdir, cd, touch

2. Вводим в файл lab8-1.asm текст программы из листинга 8.1. Создаем исполняемый файл и запускаем его.



```
lab8-1.asm      [-M--] 31 L:[ 1+11 12/ 20] *(358 / 649b) 0010 0x00A
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.2: mcedit

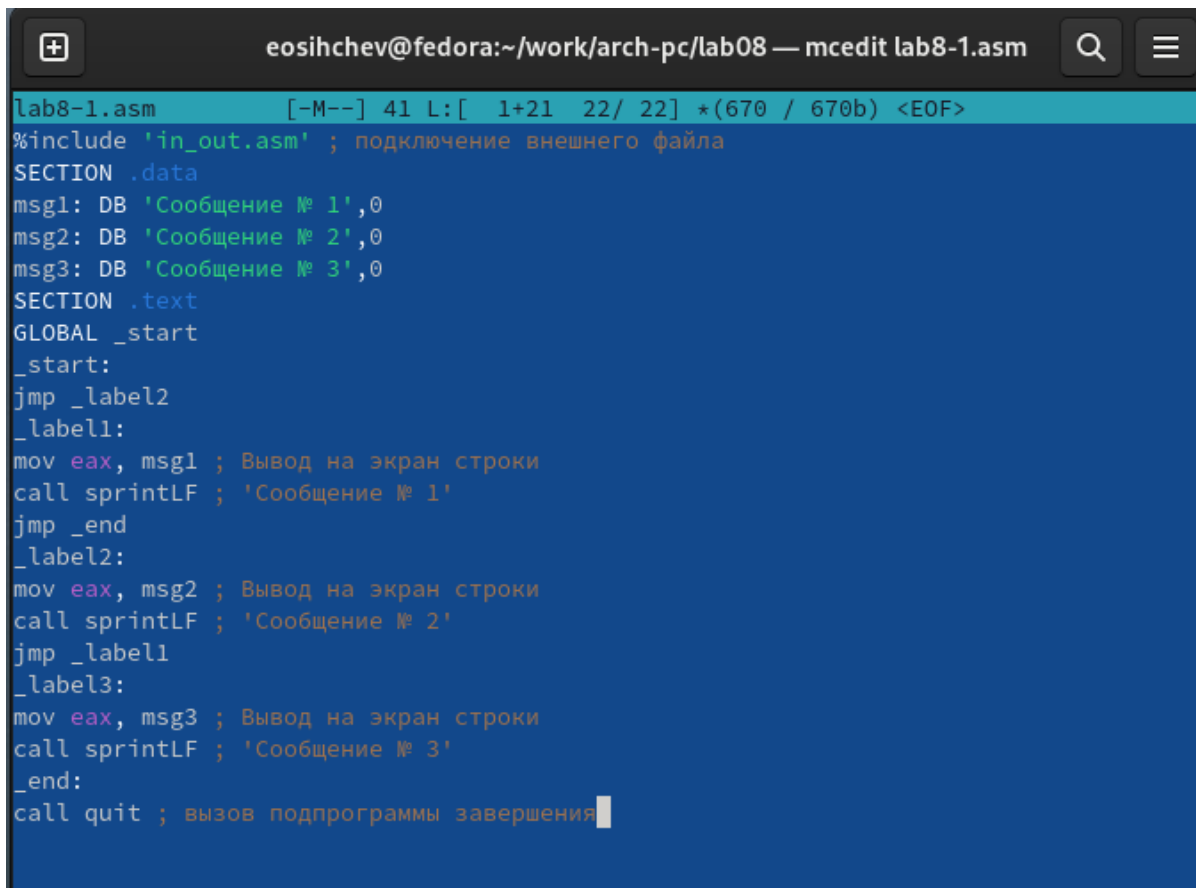


```
[eosihchev@fedora lab08]$ mcedit lab8-1.asm

[eosihchev@fedora lab08]$ nasm -f elf lab8-1.asm
[eosihchev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[eosihchev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[eosihchev@fedora lab08]$
```

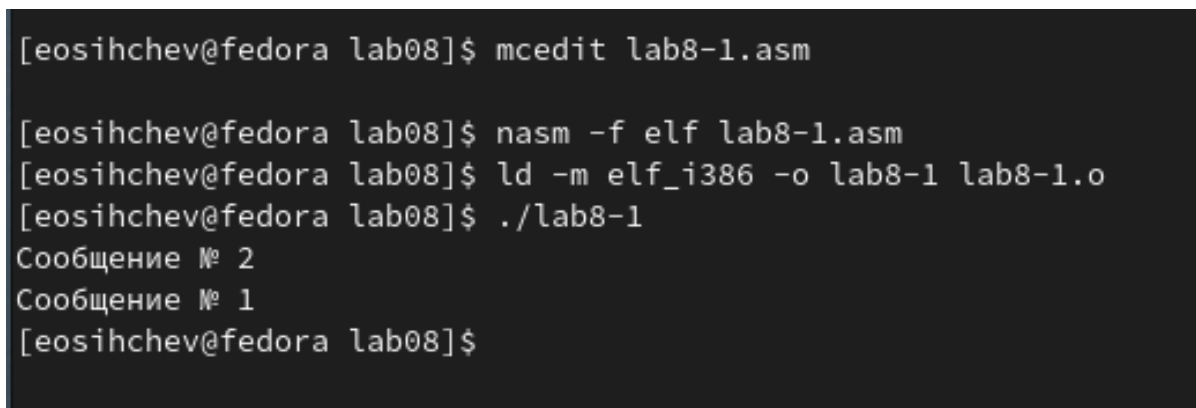
Рис. 2.3: Транслирование, компоновка, запуск

3. Изменяем текст программы в соответствии с листингом 8.2. Создаем исполняемый файл и запускаем его.



```
lab8-1.asm      [-M--] 41 L:[ 1+21 22/ 22] *(670 / 670b) <EOF>
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.4: mcedit



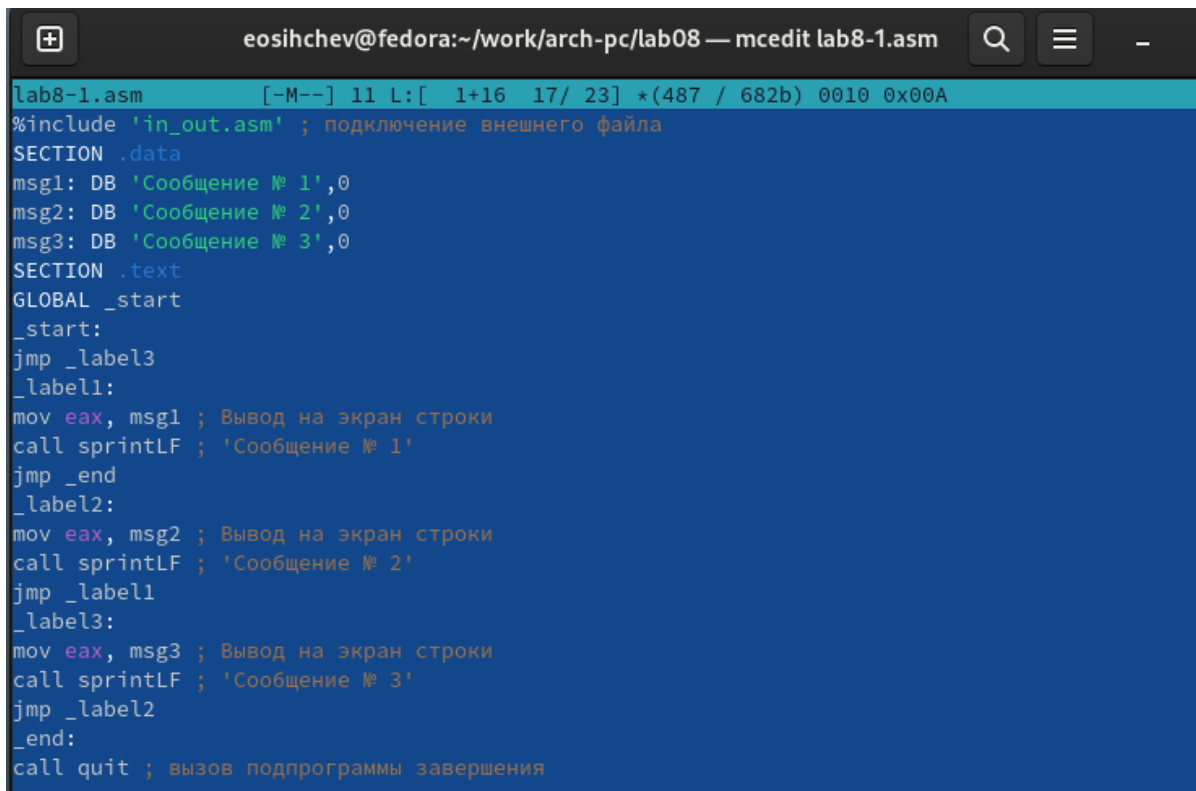
```
[eosihchev@fedora lab08]$ mcedit lab8-1.asm

[eosihchev@fedora lab08]$ nasm -f elf lab8-1.asm
[eosihchev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[eosihchev@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[eosihchev@fedora lab08]$
```

Рис. 2.5: Транслирование, компоновка, запуск

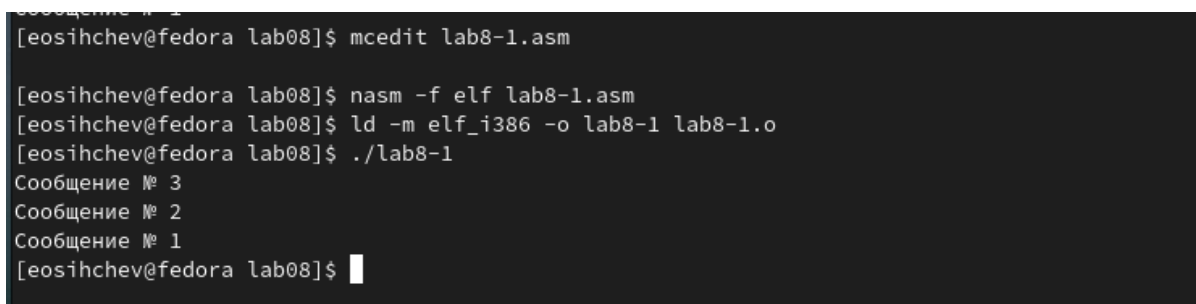
4. Изменяем текст программы так, чтобы вывод был в обратном порядке.  
Создаем исполняемый файл и запускаем его.





```
lab8-1.asm      [-M--] 11 L:[ 1+16 17/ 23] *(487 / 682b) 0010 0x00A
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
jmp _label2
_end:
call quit ; вызов подпрограммы завершения
```

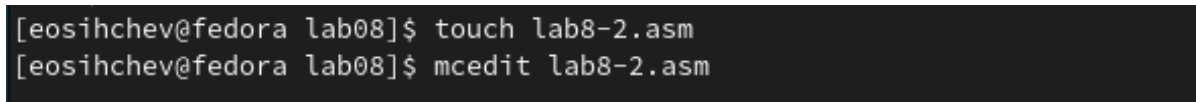
Рис. 2.6: mcedit



```
[eosihchev@fedora lab08]$ mcedit lab8-1.asm
[eosihchev@fedora lab08]$ nasm -f elf lab8-1.asm
[eosihchev@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[eosihchev@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[eosihchev@fedora lab08]$
```

Рис. 2.7: Транслирование, компоновка, запуск

5. Создаем файл lab8-2.asm и вводим в него текст программы из листинга 8.3.



```
[eosihchev@fedora lab08]$ touch lab8-2.asm
[eosihchev@fedora lab08]$ mcedit lab8-2.asm
```

Рис. 2.8: Команды: touch, mcedit

```
lab8-2.asm [-M--] 29 L: [ 1+40 41/ 49] *(1508/1743b) 0010 0x00A
%include 'in_out.asm'
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:
; ----- Вывод сообщения 'Введите B: '
mov eax,msg1
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 2.9: mcedit

6. Создаем исполняемый файл и проверяем его работу для разных значений B.

```
[eosihchev@fedora lab08]$ nasm -f elf lab8-2.asm
[eosihchev@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[eosihchev@fedora lab08]$ ./lab8-2
Введите B: 51
Наибольшее число: 51
[eosihchev@fedora lab08]$ ./lab8-2
Введите B: 49
Наибольшее число: 50
[eosihchev@fedora lab08]$ ./lab8-2
Введите B: 19
Наибольшее число: 50
[eosihchev@fedora lab08]$
```

Рис. 2.10: Транслирование, компоновка, запуск

7. Выполняем трансляцию файла lab8-2.asm с получением файла листинга и открываем его с помощью текстового редактора mcedit.

```
[eosihchev@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
[eosihchev@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 2.11: Транслирование с файлом листинга | mcedit

```
lab8-2.lst      [----] 48 L: [175+33 208/225] *(13156/14458b) 0010 0x00A
3 00000012 00.....
4 00000013 D09D0B0D0B8D0B1D0-      msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000.....
5 00000035 32300000              A dd '20'
6 00000039 35300000              C dd '50'
7                                section .bss
8 00000000 <res Ah>              max resb 10
9 0000000A <res Ah>              B resb 10
10                               section .text
11                               global _start
12                               _start:
13                               ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000]          mov eax,msg1
15 000000ED E81DFFFFFF           call sprint
16                               ; ----- Ввод 'B'
17 000000F2 B9[0A000000]          mov ecx,B
18 000000F7 BA0A000000           mov edx,10
19 000000FC E842FFFFFF           call sread
20                               ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]          mov eax,B
22 00000106 E891FFFFFF           call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000]          mov [B],eax ; запись преобразованного числа в 'B'
24                               ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]        mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000]        mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]        cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C                 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000]        mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000]        mov [max],ecx ; 'max = C'
32                               ; ----- Преобразование 'max(A,C)' из символа в число
33                               check_B:
34 00000130 B8[00000000]          mov eax,max
35 00000135 E862FFFFFF           call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000]          mov [max],eax ; запись преобразованного числа в 'max'
37                               ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000]        mov ecx,[max]
39 00000145 3B0D[0A000000]        cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 0000014B 7F0C                 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 0000014D 8B0D[0A000000]        mov ecx,[B] ; иначе 'ecx = B'
42 00000153 890D[00000000]        mov [max],ecx
43                               ; ----- Вывод результата
44                               fin:
45 00000159 B8[13000000]          mov eax, msg2
46 0000015E E8ACFFFFFF           call sprint ; Вывод сообщения 'Наибольшее число: '
47 00000163 A1[00000000]          mov eax,[max]
48 00000168 E819FFFFFF           call iprintLF ; Вывод 'max(A,B,C)'
49 0000016D E869FFFFFF           call quit ; Выход
```

Рис. 2.12: Файл lab8-2.lst

|                          |  |
|--------------------------|--|
| 46 0000015E E8ACFEFFFF   | call sprint ; Вывод сообщения 'Наибольшее число: ' |
| 47 00000163 A1[00000000] | mov eax,[max]                                      |
| 48 00000168 E819FFFFFF   | call iprintLF ; Вывод 'max(A,B,C)'                 |
| 49 0000016D E869FFFFFF   | call quit ; Выход                                  |

Рис. 2.13: Строки 46-49

Разберем строки под номерами 46,48,49:

- 46(Номер строки) 00000159(Адрес) E8B1FEFFFF(Машинный код) call sprint ; Вывод сообщения 'Наибольшее число:' (Исходный текст программы)

Инструкция call sprint начинается по смещению 00000159 в сегменте кода. Далее мы

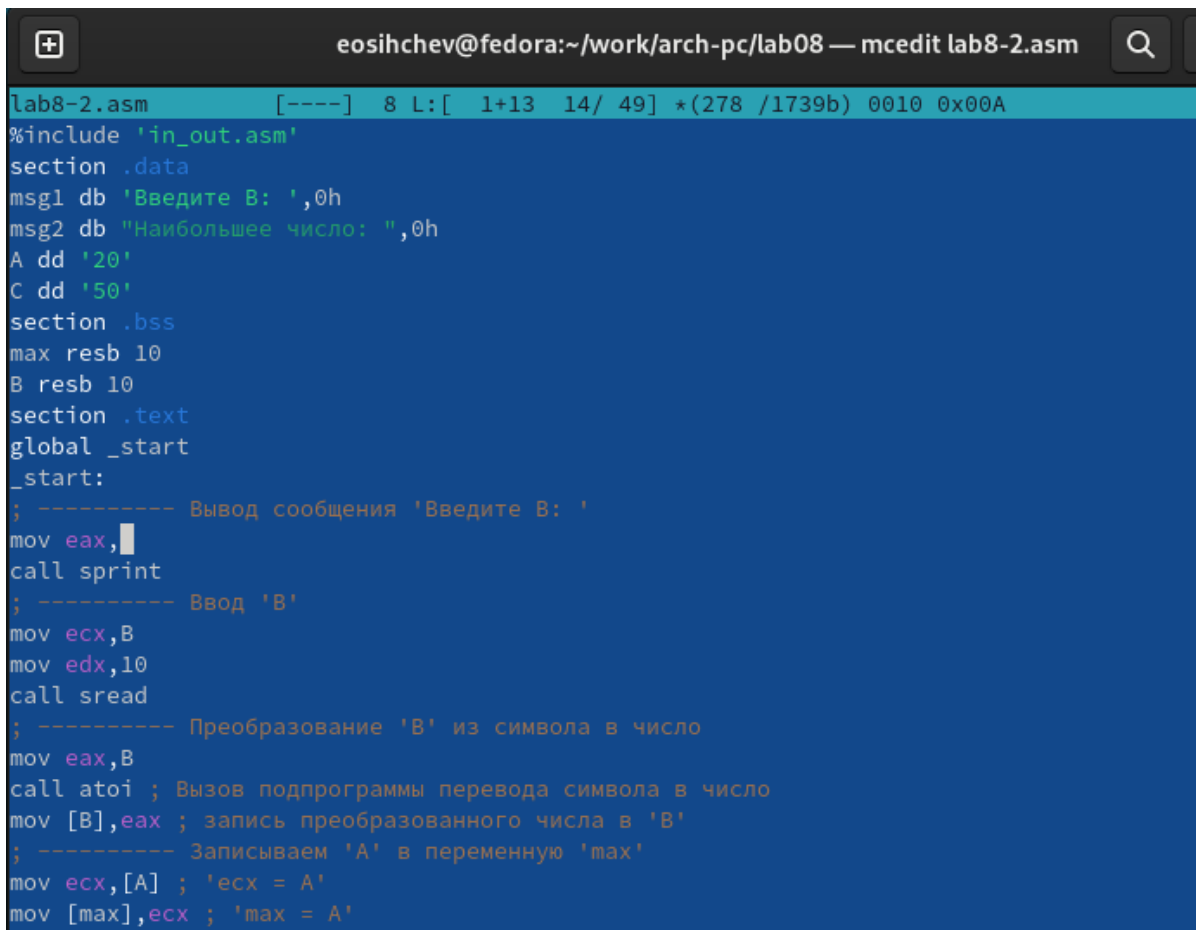
- 48(Номер строки) 00000163(Адрес) E81EFFFFFF(Машинный код) call iprintLF ; Вывод 'max(A,B,C)' (Исходный текст программы)

Инструкция call iprintLF начинается по смещению 00000163 в сегменте кода. Далее мы можем понять, что инструкция call iprintLF ассемблируется в машинный код E81EFFFFFF (в шестнадцатеричном представлении).

- 49(Номер строки) 00000168(Адрес) E86EFFFFFF(Машинный код) call quit ; Выход (Исходный текст программы)

Инструкция call quit начинается по смещению 00000168 в сегменте кода. Далее мы можем понять, что инструкция call quit ассемблируется в машинный код E86EFFFFFF (в шестнадцатеричном представлении).

8. Открываем файл с программой lab8-2.asm и в любой инструкции с двумя операндами удаляем один операнд. Выполняем трансляцию с получением файла листинга.



```
lab8-2.asm [----] 8 L: [ 1+13 14/ 49] *(278 /1739b) 0010 0x00A
#include 'in_out.asm'
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:
; ----- Вывод сообщения 'Введите B: '
mov eax,0
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A] ; 'ecx = A'
mov [max],ecx ; 'max = A'
```

Рис. 2.14: mcedit

При попытке транслирования терминал выдает ошибку.

```
[eiosihchev@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
lab8-2.asm:14: error: invalid combination of opcode and operands
```

Рис. 2.15: Транслирование с файлом листинга | Ошибка

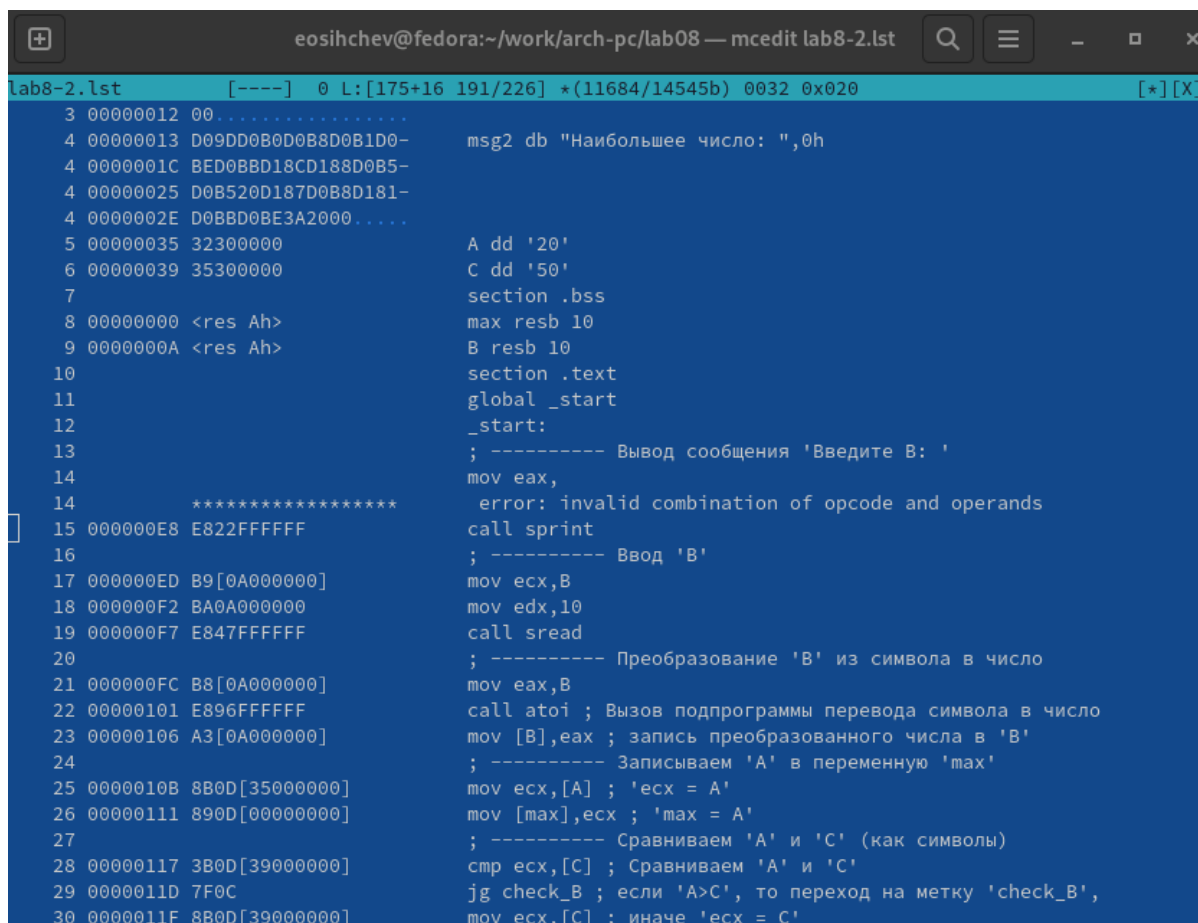
Как видим, в данном случае создается только файл листинга.

```
[eiosihchev@fedora lab08]$ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.lst
[eiosihchev@fedora lab08]$
```

Рис. 2.16: Команда ls

Можем заметить, что в файле листинга после строки, где мы убрали операнд,

появляется сообщение об ошибке (такое же как в терминале при попытке от-транслировать текст программы в объектный файл).



```
lab8-2.lst      [----]  0 L:[175+16 191/226] *(11684/14545b) 0032 0x020  [*] [X]
3 00000012 00.....
4 00000013 D09DD0B0D0B8D0B1D0-      msg2 db "Наибольшее число: ",0h
4 0000001C BED0BBD18CD188D0B5-
4 00000025 D0B520D187D0B8D181-
4 0000002E D0BBD0BE3A2000.....
5 00000035 32300000      A dd '20'
6 00000039 35300000      C dd '50'
7                                section .bss
8 00000000 <res Ah>      max resb 10
9 0000000A <res Ah>      B resb 10
10                               section .text
11                               global _start
12                               _start:
13                               ; ----- Вывод сообщения 'Введите B: '
14                               mov eax,
14                               *****      error: invalid combination of opcode and operands
15 000000E8 E822FFFFFF      call sprint
16                               ; ----- Ввод 'B'
17 000000ED B9[0A000000]      mov ecx,B
18 000000F2 BA0A000000      mov edx,10
19 000000F7 E847FFFFFF      call sread
20                               ; ----- Преобразование 'B' из символа в число
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
23 00000106 A3[0A000000]      mov [B],eax ; запись преобразованного числа в 'B'
24                               ; ----- Записываем 'A' в переменную 'max'
25 0000010B 8B0D[35000000]      mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]      mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 00000117 3B0D[39000000]      cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 0000011D 7F0C      jg check_B ; если 'A>C', то переход на метку 'check_B',
30 0000011F 8B0D[39000000]      mov ecx,[C] ; иначе 'ecx = C'
```

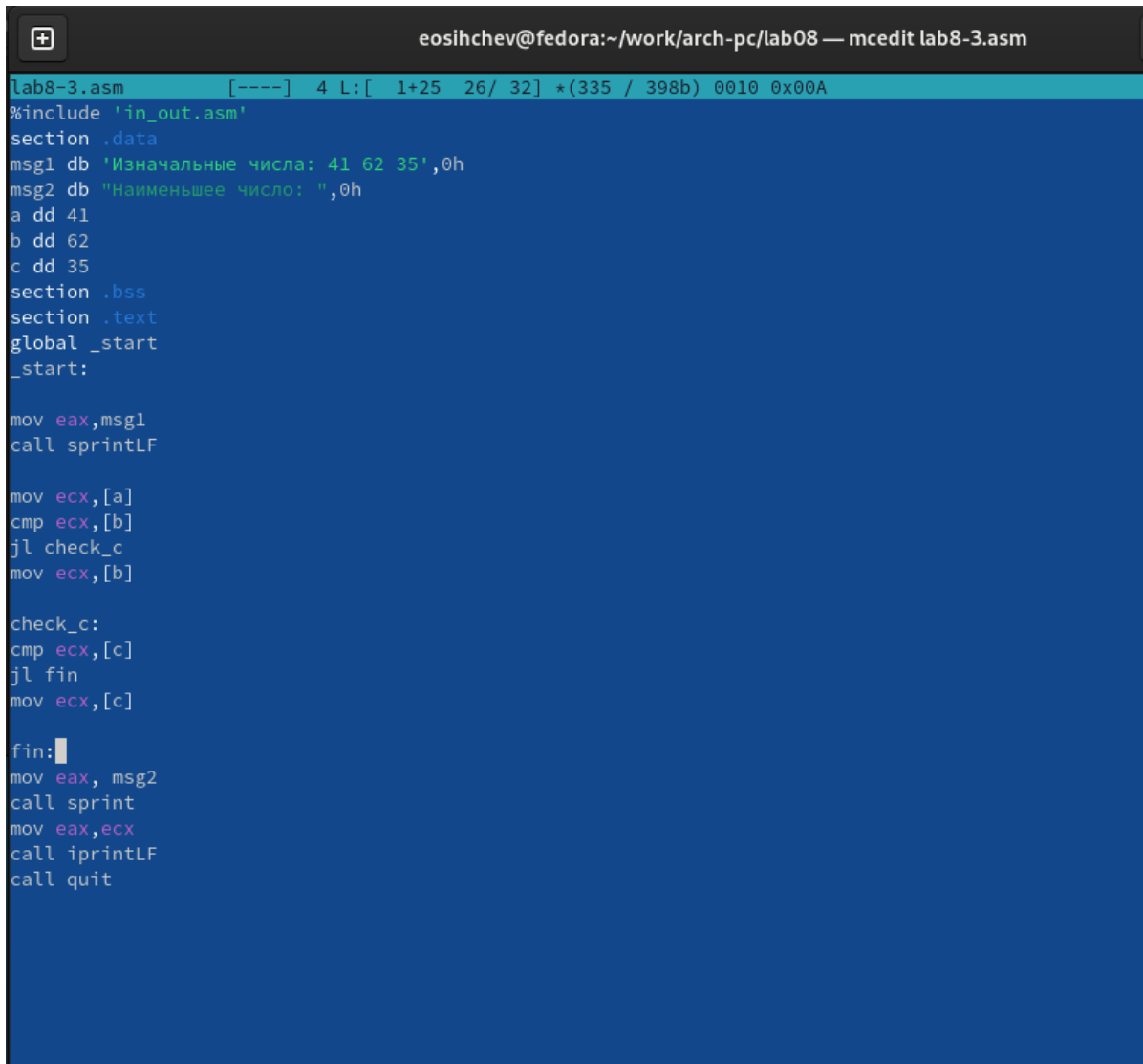
Рис. 2.17: Файл lab8-2.lst

### 3 Самостоятельная работа

1. Создаем файл lab8-3.asm и вводим в него текст программы для нахождения наименьшего числа из 41,62,35 (Вариант №10). Создаем исполняемый файл и запускаем его.

```
[eosihchev@fedora ~]$ cd /work/aren-pe/lab08  
[eosihchev@fedora lab08]$ touch lab8-3.asm  
[eosihchev@fedora lab08]$ mcedit lab8-3.asm
```

Рис. 3.1: Команды: touch, mcedit



```
eosihchev@fedora:~/work/arch-pc/lab08 — mcedit lab8-3.asm
lab8-3.asm [----] 4 L: [ 1+25 26/ 32] *(335 / 398b) 0010 0x00A
#include 'in_out.asm'
section .data
msg1 db 'Изначальные числа: 41 62 35',0h
msg2 db "Наименьшее число: ",0h
a dd 41
b dd 62
c dd 35
section .bss
section .text
global _start
_start:

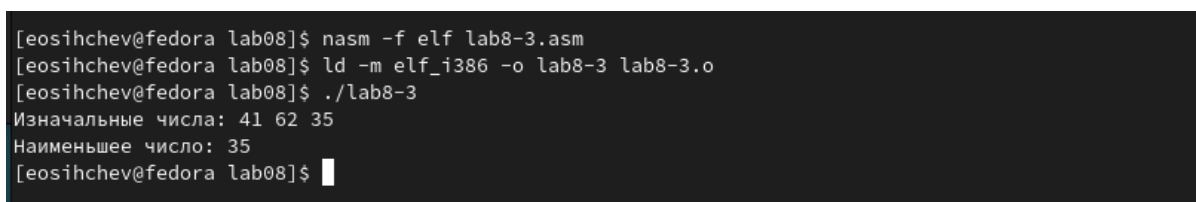
mov eax,msg1
call sprintLF

mov ecx,[a]
cmp ecx,[b]
jl check_c
mov ecx,[b]

check_c:
cmp ecx,[c]
jl fin
mov ecx,[c]

fin:
mov eax, msg2
call sprint
mov eax,ecx
call iprintLF
call quit
```

Рис. 3.2: mcedit



```
[eosihchev@fedora lab08]$ nasm -f elf lab8-3.asm
[eosihchev@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[eosihchev@fedora lab08]$ ./lab8-3
Изначальные числа: 41 62 35
Наименьшее число: 35
[eosihchev@fedora lab08]$
```

Рис. 3.3: Транслирование, компоновка, запуск

2. Создаем файл lab8-4.asm и вводим в него текст программы, которая для введенных с клавиатуры значений x и a вычисляет значение заданной



функции  $f(x)$ . Создаем исполняемый файл и запускаем его.

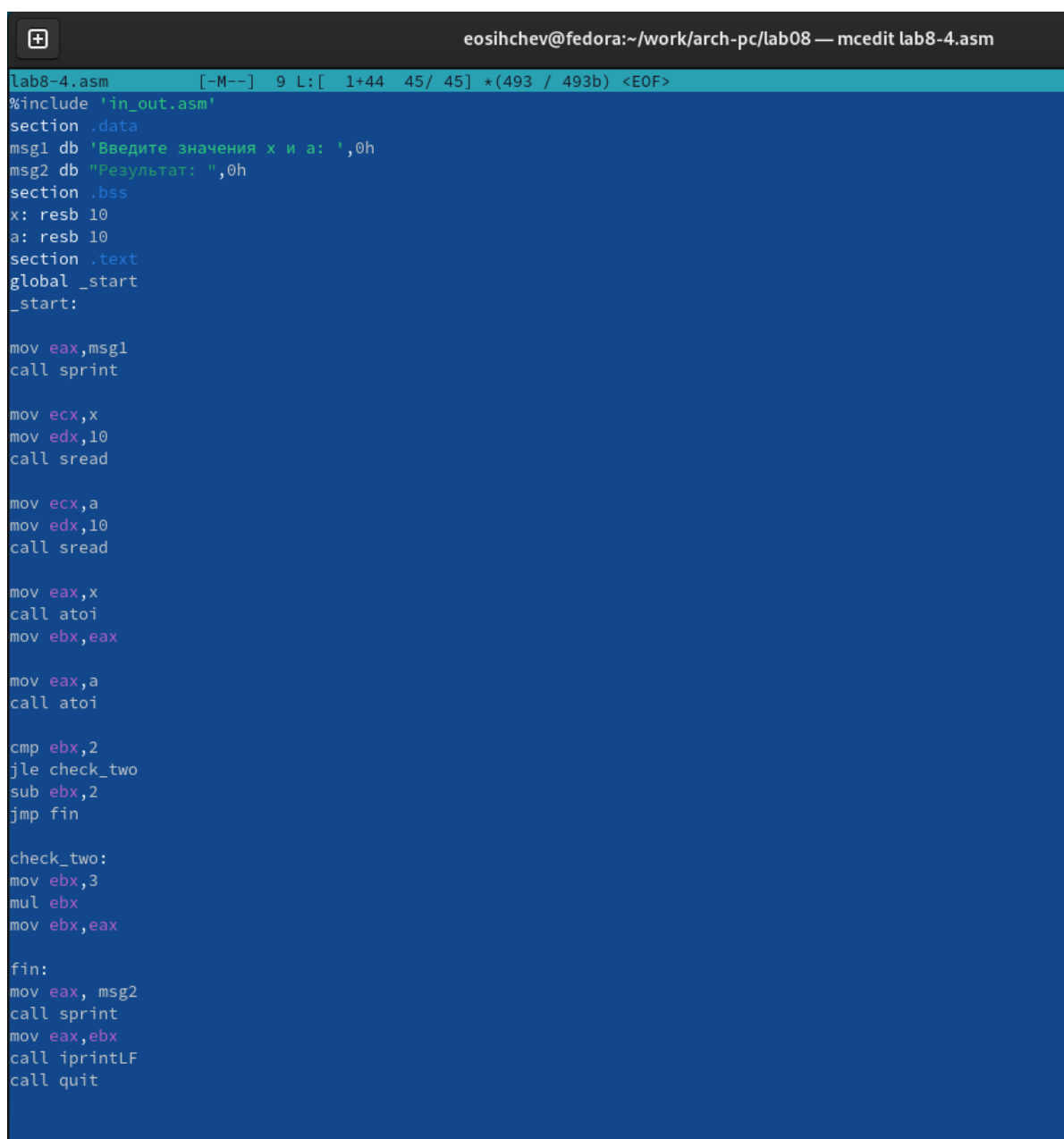
Функция  $f(x)$  и значения  $x, a$  для проверки:

**10**      
$$\begin{cases} x - 2, & x > 2 \\ 3a, & x \leq 2 \end{cases} \quad (3;0) \quad (1;2)$$

Рис. 3.4: Функция и значения  $x, a$  для проверки

```
[eosihchev@fedora lab08]$ touch lab8-4.asm  
[eosihchev@fedora lab08]$ mcedit lab8-4.asm
```

Рис. 3.5: Команды: touch, mcedit



```
lab8-4.asm      [-M--]  9 L: [ 1+44  45/ 45] *(493 / 493b) <EOF>
#include 'in_out.asm'
section .data
msg1 db 'Введите значения x и a: ',0h
msg2 db "Результат: ",0h
section .bss
x: resb 10
a: resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,x
mov edx,10
call sread

mov ecx,a
mov edx,10
call sread

mov eax,x
call atoi
mov ebx,eax

mov eax,a
call atoi

cmp ebx,2
jle check_two
sub ebx,2
jmp fin

check_two:
mov ebx,3
mul ebx
mov ebx,eax

fin:
mov eax, msg2
call sprint
mov eax,ebx
call iprintLF
call quit
```

Рис. 3.6: mcedit

```
[eosihchev@fedora lab08]$ nasm -f elf lab8-4.asm
[eosihchev@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[eosihchev@fedora lab08]$ ./lab8-4
Введите значения x и a: 3
0
Результат: 1
[eosihchev@fedora lab08]$ ./lab8-4
Введите значения x и a: 1
2
Результат: 6
[eosihchev@fedora lab08]$
```

Рис. 3.7: Транслирование, компоновка, запуск

## 4 Вывод

Изучил команды условного и безусловного переходов. Приобрел навыки написания программ с использованием переходов. Ознакомился с назначением и структурой файла листинга.