

Отчёта по лабораторной работе 8

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Грязнов Михаил Александрович НПИбд-01-22

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	22

Список иллюстраций

2.1	Файл lab8-1.asm:	7
2.2	Программа lab8-1.asm:	8
2.3	Файл lab8-1.asm:	9
2.4	Программа lab8-1.asm:	10
2.5	Файл lab8-1.asm	11
2.6	Программа lab8-1.asm	12
2.7	Файл lab8-2.asm	13
2.8	Программа lab8-2.asm	14
2.9	Файл листинга lab8-2	15
2.10	ошибка трансляции lab8-2	16
2.11	файл листинга с ошибкой lab8-2	17
2.12	Файл lab8-3.asm	18
2.13	Программа lab8-3.asm	19
2.14	Файл lab8-4.asm	20
2.15	Программа lab8-4.asm	21

Список таблиц

1 Цель работы

Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Введите в файл lab8-1.asm текст программы из листинга 8.1. (рис. 2.1)

```
lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

%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

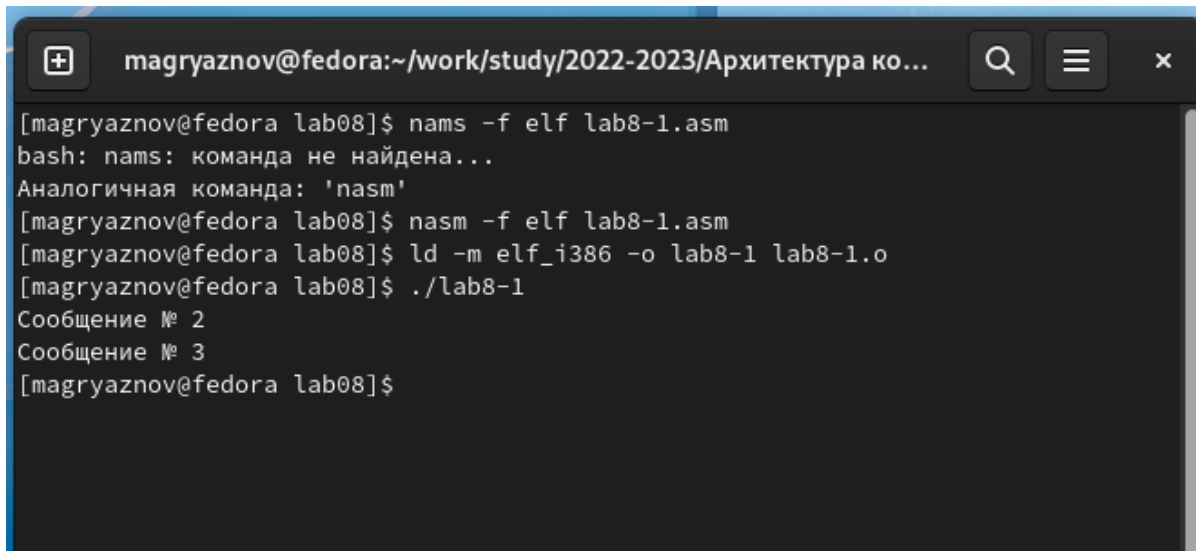
_label1:
mov eax, msg1 ; Вывод на экран строки
call printf ; 'Сообщение № 1'

_label2:
mov eax, msg2 ; Вывод на экран строки
call printf ; 'Сообщение № 2'

_label3:
mov eax, msg3 ; Вывод на экран строки
call printf ; 'Сообщение № 3'
|
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.1: Файл lab8-1.asm:

Создайте исполняемый файл и запустите его. (рис. 2.2)



```
magryaznov@fedora:~/work/study/2022-2023/Архитектура ко...
[magryaznov@fedora lab08]$ nams -f elf lab8-1.asm
bash: nams: команда не найдена...
Аналогичная команда: 'nasm'
[magryaznov@fedora lab08]$ nasm -f elf lab8-1.asm
[magryaznov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[magryaznov@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[magryaznov@fedora lab08]$
```

Рис. 2.2: Программа lab8-1.asm:

Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Измените текст программы в соответствии с листингом 8.2. (рис. 2.3, 2.4)

Открыть ▾  lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

```
%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

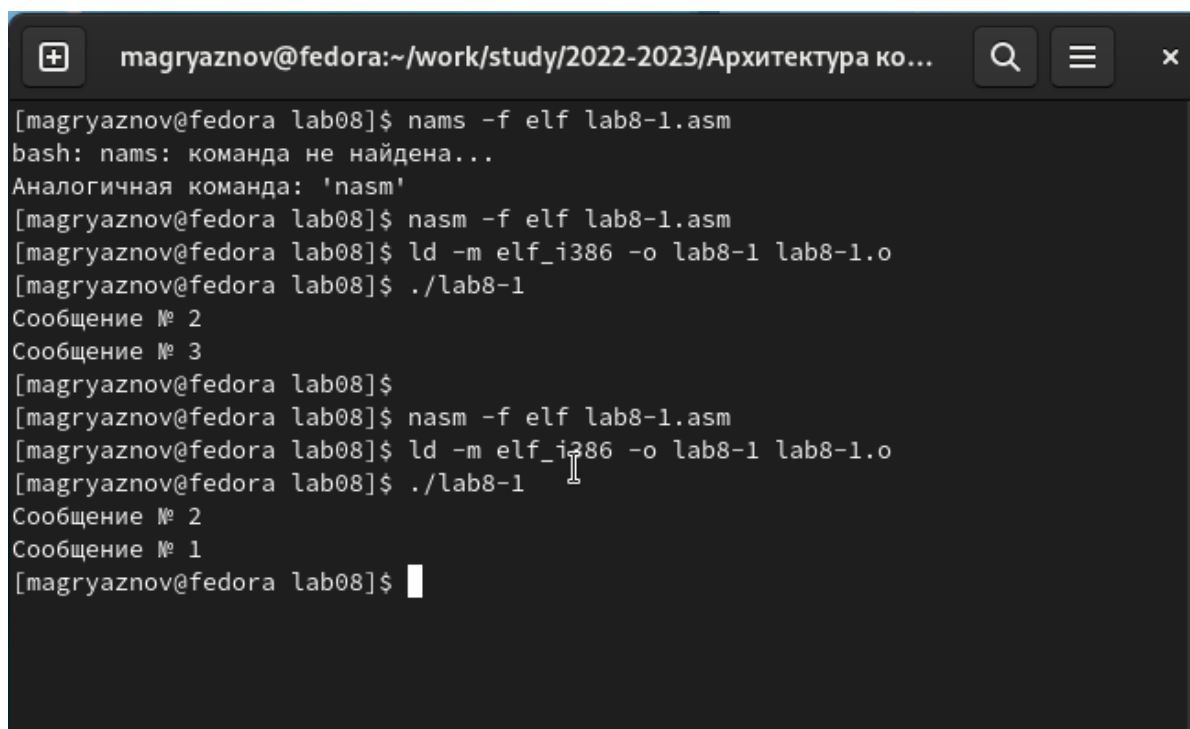
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.3: Файл lab8-1.asm:



```
magryaznov@fedora:~/work/study/2022-2023/Архитектура ко...
[magryaznov@fedora lab08]$ nasm -f elf lab8-1.asm
bash: nasm: команда не найдена...
Аналогичная команда: 'nasm'
[magryaznov@fedora lab08]$ nasm -f elf lab8-1.asm
[magryaznov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[magryaznov@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
[magryaznov@fedora lab08]$
[magryaznov@fedora lab08]$ nasm -f elf lab8-1.asm
[magryaznov@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[magryaznov@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[magryaznov@fedora lab08]$
```

Рис. 2.4: Программа lab8-1.asm:

Измените текст программы добавив или изменив инструкции `jmp`, чтобы вывод программы был следующим (рис. 2.5, 2.6):

Сообщение № 3

Сообщение № 2

Сообщение № 1

Открыть ▾ 

lab8-1.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

```
%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. 2.5: Файл lab8-1.asm

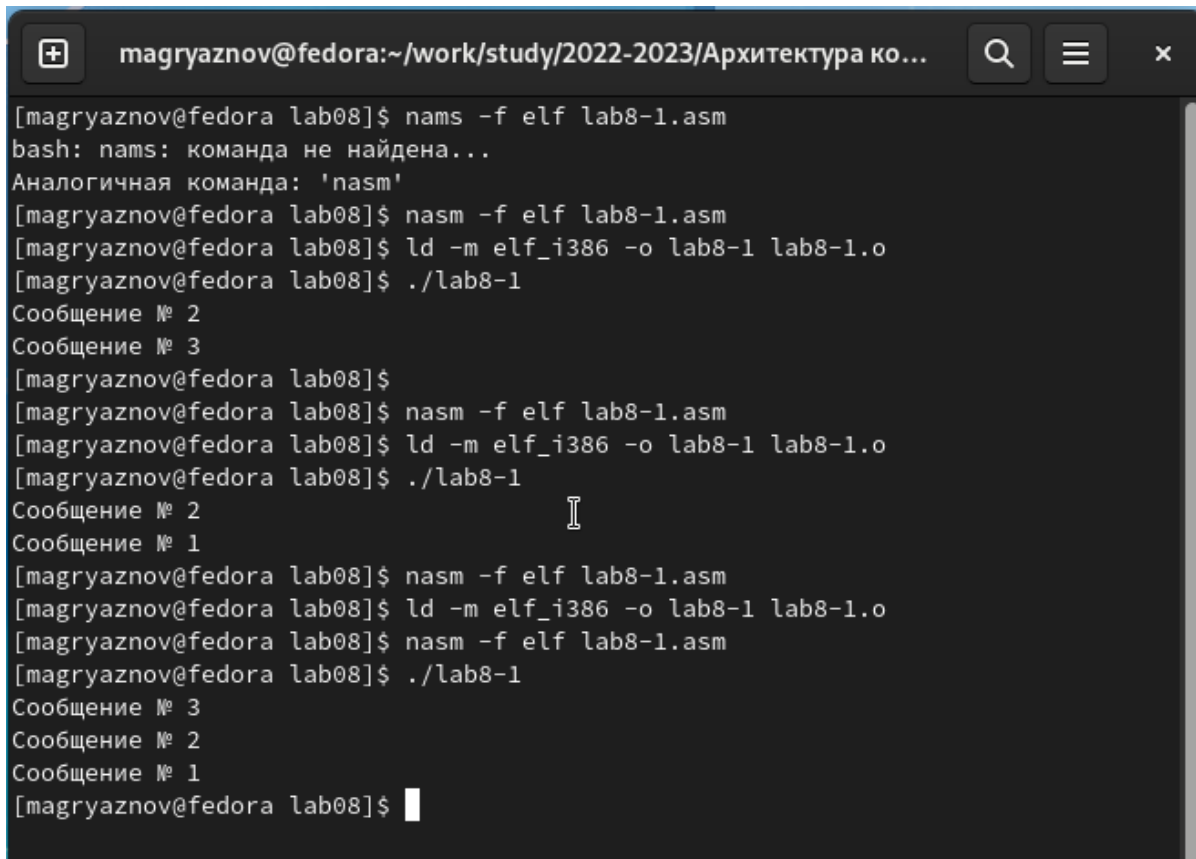
A terminal window titled 'magryaznov@fedora:~/work/study/2022-2023/Архитектура ко...' with search, menu, and close icons. The terminal shows a sequence of commands and messages. The user attempts to run 'nasm -f elf lab8-1.asm', which fails with 'bash: nasm: команда не найдена...'. They then run 'nasm -f elf lab8-1.asm' successfully. Next, they run 'ld -m elf_i386 -o lab8-1 lab8-1.o' and './lab8-1'. This is followed by three messages: 'Сообщение № 2', 'Сообщение № 3', and 'Сообщение № 1'. The user repeats the assembly and linking commands, followed by another 'Сообщение № 2' and 'Сообщение № 1'. Finally, they run the program again, resulting in 'Сообщение № 3', 'Сообщение № 2', and 'Сообщение № 1'.

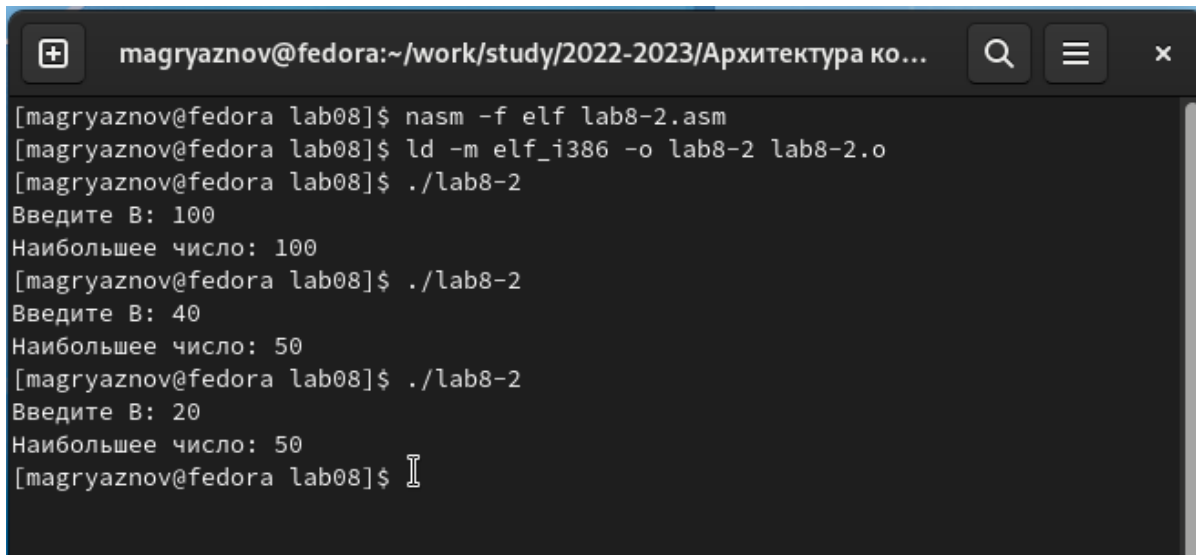
Рис. 2.6: Программа lab8-1.asm

3. Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А, В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры. Создайте исполняемый файл и проверьте его работу для разных значений В. (рис. 2.7, 2.8)

Открыть + lab8-2.asm ~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

```
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'
```

Рис. 2.7: Файл lab8-2.asm

A terminal window with a dark background and light text. The window title is "magryaznov@fedora:~/work/study/2022-2023/Архитектура ко...". The terminal shows the following commands and output:

```
[magryaznov@fedora lab08]$ nasm -f elf lab8-2.asm
[magryaznov@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[magryaznov@fedora lab08]$ ./lab8-2
Введите В: 100
Наибольшее число: 100
[magryaznov@fedora lab08]$ ./lab8-2
Введите В: 40
Наибольшее число: 50
[magryaznov@fedora lab08]$ ./lab8-2
Введите В: 20
Наибольшее число: 50
[magryaznov@fedora lab08]$
```

Рис. 2.8: Программа lab8-2.asm

4. Обычно `nasm` создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ `-l` и задав имя файла листинга в командной строке. Создайте файл листинга для программы из файла `lab8-2.asm` (рис. 2.9)

```
43 <1>
44 <1> ;----- sprintf -----
45 <1> ; Функция печати сообщения с переводом строки
46 <1> ; входные данные: mov eax,<message>
47 <1> sprintf:
48 0000002D E8DDFFFFFF <1> call sprint
49 <1>
50 00000032 50 <1> push eax
51 00000033 B80A000000 <1> mov eax, 0AH
52 00000038 50 <1> push eax
53 00000039 89E0 <1> mov eax, esp
54 0000003B E8CFFFFFFF <1> call sprint
55 00000040 58 <1> pop eax
56 00000041 58 <1> pop eax
57 00000042 C3 <1> ret
58 <1>
59 <1> ;----- sread -----
60 <1> ; Функция считывания сообщения
61 <1> ; входные данные: mov eax,<buffer>, mov ebx,<N>
62 <1> sread:
63 00000043 53 <1> push ebx
64 00000044 50 <1> push eax
65 <1>
66 00000045 BB00000000 <1> mov ebx, 0
67 0000004A B803000000 <1> mov eax, 3
68 0000004F CD80 <1> int 80h
69 <1>
70 00000051 5B <1> pop ebx
71 00000052 59 <1> pop ecx
```

Рис. 2.9: Файл листинга lab8-2

Внимательно ознакомиться с его форматом и содержимым. Подробно объяснить содержимое трёх строк файла листинга по выбору.

строка 10

- 10 - номер строки
- 00000006 - адрес
- 7403 - машинный код
- jz finished - код программы

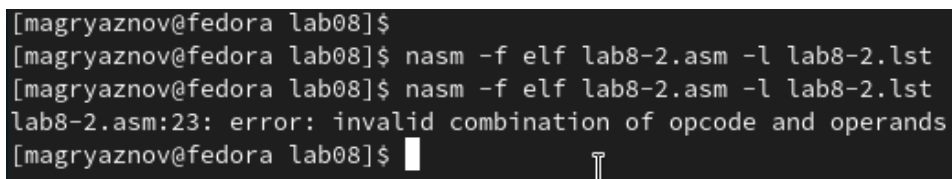
строка 11

- 11 - номер строки
- 00000008 - адрес
- 40 - машинный код
- inc eax - код программы

строка 12

- 12 - номер строки
- 00000009 - адрес
- EBF8 - машинный код
- jmp nextchar - код программы

Откройте файл с программой lab8-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните трансляцию с получением файла листинга (рис. 2.10, 2.11)

A screenshot of a terminal window with a dark background. The prompt is [magryaznov@fedora lab08]\$. The user enters 'nasm -f elf lab8-2.asm -l lab8-2.lst' twice. The second time, an error message appears: 'lab8-2.asm:23: error: invalid combination of opcode and operands'. The cursor is at the end of the command line.

```
[magryaznov@fedora lab08]$  
[magryaznov@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst  
[magryaznov@fedora lab08]$ nasm -f elf lab8-2.asm -l lab8-2.lst  
lab8-2.asm:23: error: invalid combination of opcode and operands  
[magryaznov@fedora lab08]$
```

Рис. 2.10: ошибка трансляции lab8-2


```

lab8-2.asm
12
13
14 000000E8 B8[00000000]
15 000000ED E81DFFFFFF
16
17 000000F2 B9[0A000000]
18 000000F7 BA0A000000
19 000000FC E842FFFFFF
20
21 00000101 B8[0A000000]
22 00000106 E891FFFFFF
23
23 *****
24
25 0000010B 8B0D[35000000]
26 00000111 890D[00000000]
27
28 00000117 3B0D[39000000]
29 0000011D 7F0C
30 0000011F 8B0D[39000000]
31 00000125 890D[00000000]
32
число
33
34 0000012B B8[00000000]
35 00000130 E867FFFFFF
36 00000135 A3[00000000]
37
38 0000013A 8B0D[00000000]
39 00000140 3B0D[0A000000]

lab8-2.lst
_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], ; запись преобразованного числа в 'B'
error: invalid combination of opcode and operands
; ----- 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'

```

Рис. 2.11: файл листинга с ошибкой lab8-2

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 8.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу (рис. 2.12,2.13)

для варианта 5 - 54, 62, 87

Открыть ▾ 

lab8-3.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

```
mov edx,80
call sread
mov eax,C
call atoi
mov [C],eax
;-----algorithm-----

mov ecx,[A] ;ecx = A
mov [min],ecx ;min = A

cmp ecx, [B] ; A&B
jnl check_C ; if a<b: goto check_C
mov ecx, [B]
mov [min], ecx ;else min = B

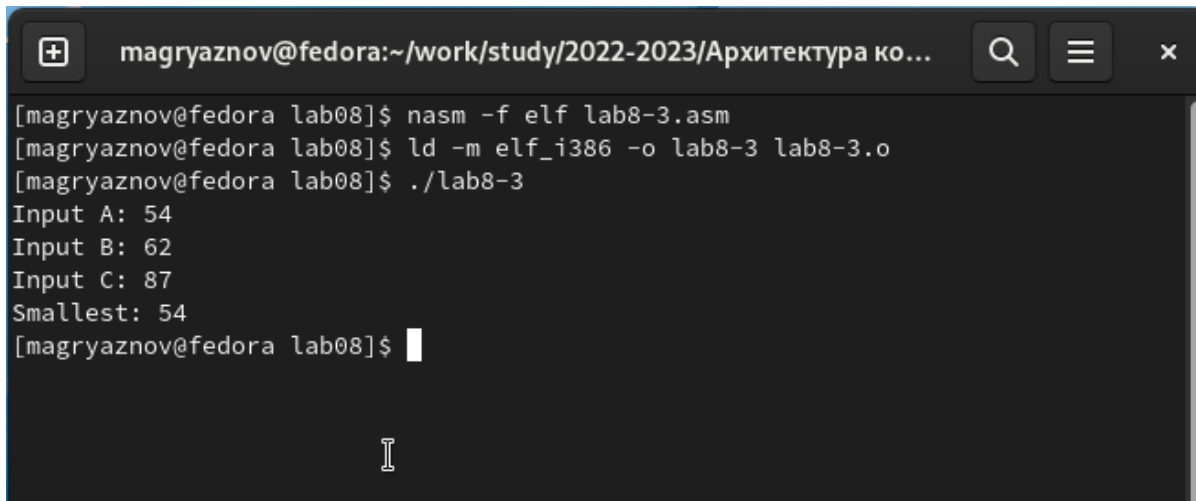
check_C:
cmp ecx, [C]
jnl finish
mov ecx,[C]
mov [min],ecx

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit
```

Рис. 2.12: Файл lab8-3.asm



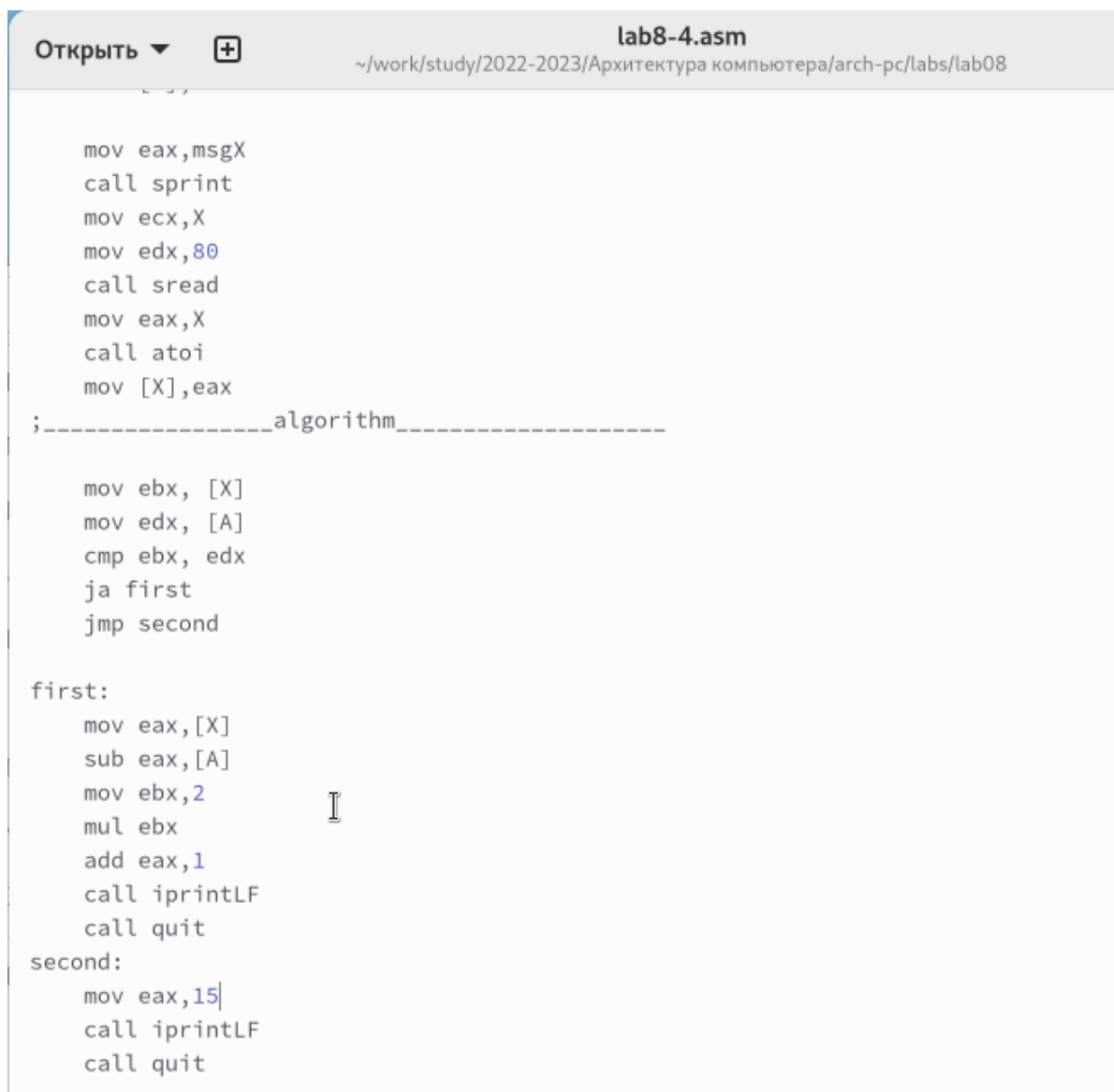
```
magryaznov@fedora:~/work/study/2022-2023/Архитектура ко...  
[magryaznov@fedora lab08]$ nasm -f elf lab8-3.asm  
[magryaznov@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o  
[magryaznov@fedora lab08]$ ./lab8-3  
Input A: 54  
Input B: 62  
Input C: 87  
Smallest: 54  
[magryaznov@fedora lab08]$
```

Рис. 2.13: Программа lab8-3.asm

6. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 8.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 8.6. (рис. 2.14, 2.15)

для варианта 5

$$\begin{cases} 2(x - a), x < a \\ 15, x > a \end{cases}$$



```
lab8-4.asm
~/work/study/2022-2023/Архитектура компьютера/arch-pc/labs/lab08

mov eax,msgX
call sprint
mov ecx,X
mov edx,80
call sread
mov eax,X
call atoi
mov [X],eax
;-----algorithm-----

mov ebx, [X]
mov edx, [A]
cmp ebx, edx
ja first
jmp second

first:
mov eax,[X]
sub eax,[A]
mov ebx,2
mul ebx
add eax,1
call iprintLF
call quit
second:
mov eax,15
call iprintLF
call quit
```

Рис. 2.14: Файл lab8-4.asm

```
[magryaznov@fedora lab08]$  
[magryaznov@fedora lab08]$ nasm -f elf lab8-4.asm  
[magryaznov@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o  
[magryaznov@fedora lab08]$ ./lab8-4  
Input A: 2  
Input X: 1  
15  
[magryaznov@fedora lab08]$ ./lab8-4  
Input A: 1  
Input X: 2  
3  
[magryaznov@fedora lab08]$
```

Рис. 2.15: Программа lab8-4.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.