

Отчёт по лабораторной работе 7

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

Ахмед МД Булбул

Содержание

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

Список иллюстраций

2.1	Программа lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	8
2.3	Программа lab7-1.asm	9
2.4	Запуск программы lab7-1.asm	10
2.5	Программа lab7-1.asm	11
2.6	Запуск программы lab7-1.asm	12
2.7	Программа lab7-2.asm	13
2.8	Запуск программы lab7-2.asm	14
2.9	Файл листинга lab7-2	15
2.10	Ошибка трансляции lab7-2	16
2.11	Файл листинга с ошибкой lab7-2	17
2.12	Программа lab7-3.asm	18
2.13	Запуск программы lab7-3.asm	18
2.14	Программа lab7-4.asm	20
2.15	Запуск программы lab7-4.asm	21

Список таблиц

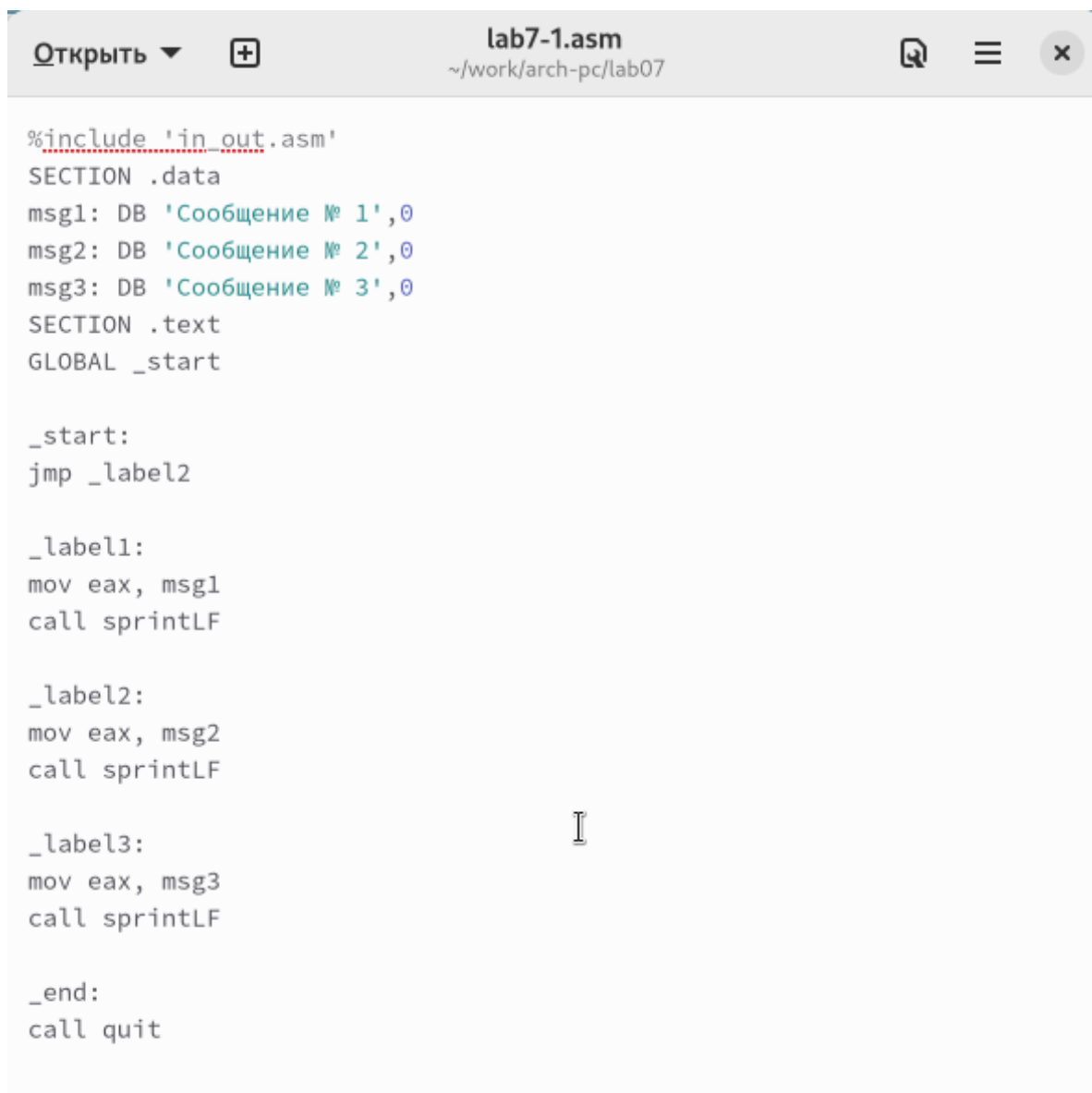
1 Цель работы

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. Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm
2. Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp.

Написал в файл lab7-1.asm текст программы из листинга 7.1.



```
lab7-1.asm
~/work/arch-pc/lab07

%include 'in_out.asm'
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
call sprintLF

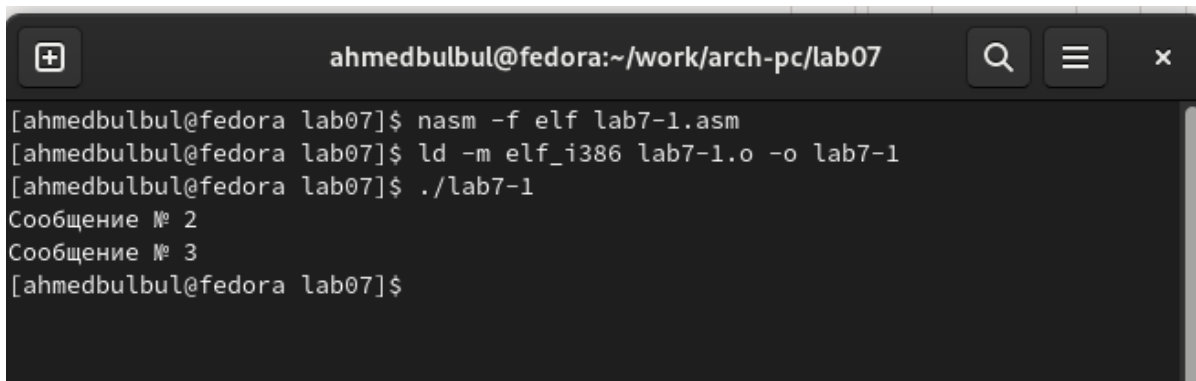
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.1: Программа lab7-1.asm

Создал исполняемый файл и запустил его.

A terminal window with a dark background. The title bar shows the user 'ahmedbulbul@fedora' and the directory '~/work/arch-pc/lab07'. The terminal contains the following text:

```
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-1.asm
[ahmedbulbul@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ahmedbulbul@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[ahmedbulbul@fedora lab07]$
```

Рис. 2.2: Запуск программы lab7-1.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`).

Изменил текст программы в соответствии с листингом 7.2.



```
%include 'in_out.asm'
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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.3: Программа lab7-1.asm

```
[ahmedbulbul@fedora lab07]$  
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-1.asm  
[ahmedbulbul@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1  
[ahmedbulbul@fedora lab07]$ ./lab7-1  
Сообщение № 2  
Сообщение № 1  
[ahmedbulbul@fedora lab07]$
```

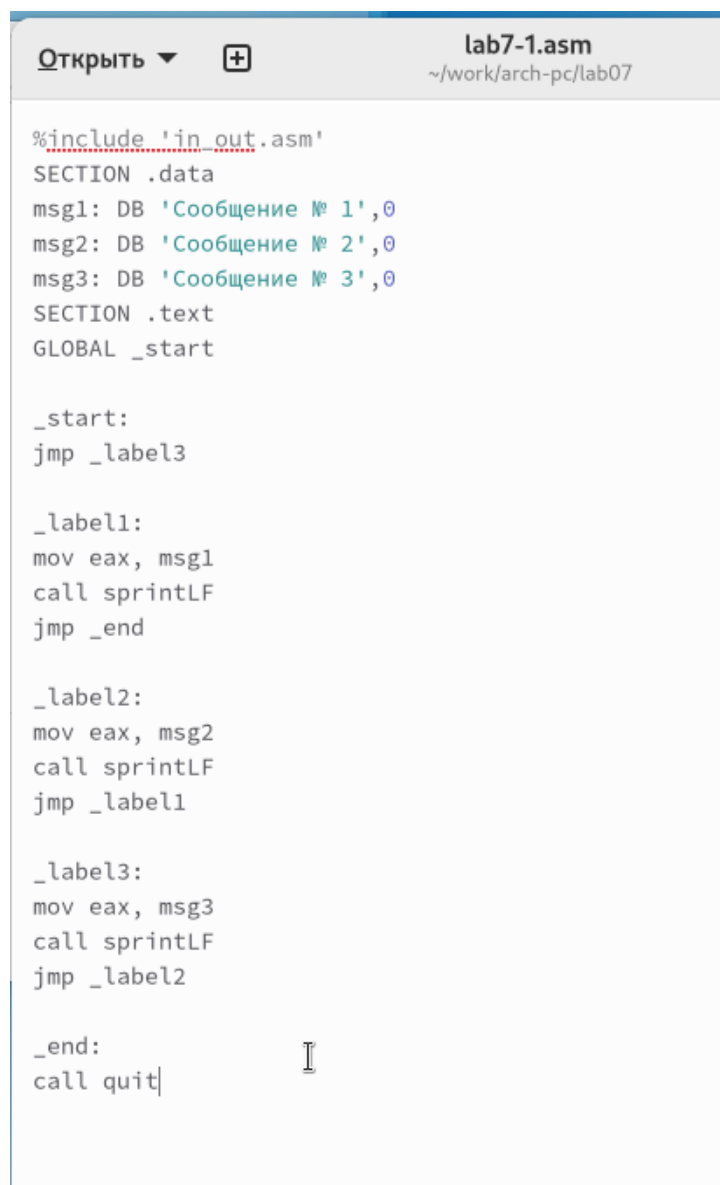
Рис. 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции `jmp`, чтобы вывод программы был следующим:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
%include 'in_out.asm'
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
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

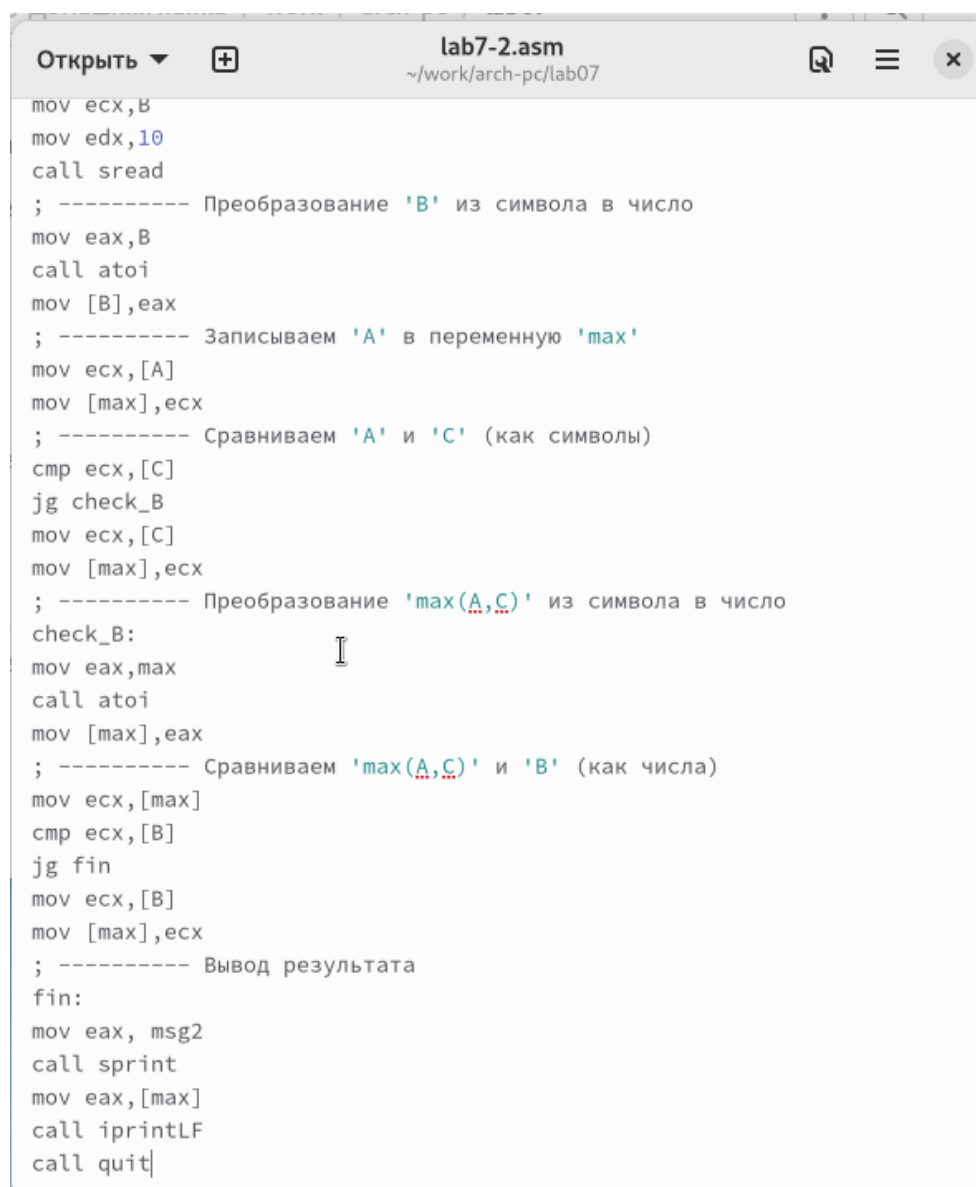
Рис. 2.5: Программа lab7-1.asm

```
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-1.asm
[ahmedbulbul@fedora lab07]$ ld -m elf_i386 lab7-1.o -o lab7-1
[ahmedbulbul@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[ahmedbulbul@fedora lab07]$
```

Рис. 2.6: Запуск программы lab7-1.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: A, B и C. Значения для A и C задаются в программе, значение B вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для разных значений B.



```
Открыть ▾ + lab7-2.asm ~\work\arch-pc\lab07
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
call quit
```

Рис. 2.7: Программа lab7-2.asm

```
[ahmedbulbul@fedora lab07]$  
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-2.asm  
[ahmedbulbul@fedora lab07]$ ld -m elf_i386 lab7-2.o -o lab7-2  
[ahmedbulbul@fedora lab07]$ ./lab7-2  
Введите В: 40  
Наибольшее число: 50  
[ahmedbulbul@fedora lab07]$ ./lab7-2  
Введите В: 70  
Наибольшее число: 70  
[ahmedbulbul@fedora lab07]$
```

Рис. 2.8: Запуск программы lab7-2.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.

Создал файл листинга для программы из файла lab7-2.asm

```

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
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 00000124 8B0D[39000000] mov ecx,[C]
31 0000012A 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi
36 0000013A A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013F 8B0D[00000000] mov ecx,[max]
39 00000145 3B0D[0A000000] cmp ecx,[B]
40 0000014B 7F0C jg fin
41 0000014D 8B0D[0A000000] mov ecx,[B]
42 00000153 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000159 B8[13000000] mov eax,msg2
46 0000015E E8ACFEFFFF call sprint

```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

строка 34

- 34 - номер строки
- 00000130 - адрес
- B8[00000000] - машинный код
- mov eax,max - код программы

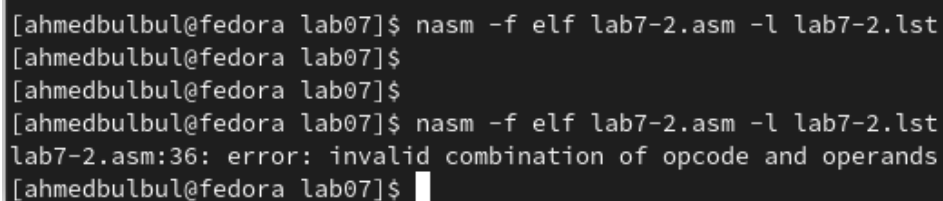
строка 35

- 35 - номер строки
- 00000135 - адрес
- E862FFFFFF - машинный код
- call atoi- код программы

строка 36

- 36 - номер строки
- 0000013A - адрес
- A3[00000000] - машинный код
- mov [max],eax - код программы

Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга.



```
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst
[ahmedbulbul@fedora lab07]$
[ahmedbulbul@fedora lab07]$
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:36: error: invalid combination of opcode and operands
[ahmedbulbul@fedora lab07]$
```

Рис. 2.10: Ошибка трансляции lab7-2


```

lab7-2.asm                                lab7-2.lst
21 00000101 B8[0A000000]                   mov eax,B
22 00000106 E891FFFFFF                   call atoi
23 0000010B A3[0A000000]                   mov [B],eax
24                                           ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]                 mov ecx,[A]
26 00000116 890D[00000000]                 mov [max],ecx
27                                           ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]                 cmp ecx,[C]
29 00000122 7F0C                           jg check_B
30 00000124 8B0D[39000000]                 mov ecx,[C]
31 0000012A 890D[00000000]                 mov [max],ecx
32                                           ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000]                   mov eax,max
35 00000135 E862FFFFFF                   call atoi
36                                           mov [max],
36 *****                                error: invalid combination of opcode and operands
37                                           ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000]                 mov ecx,[max]
39 00000140 3B0D[0A000000]                 cmp ecx,[B]
40 00000146 7F0C                           jg fin
41 00000148 8B0D[0A000000]                 mov ecx,[B]
42 0000014E 890D[00000000]                 mov [max],ecx
43                                           ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000]                   mov eax,msg2
46 00000159 E8B1FFFFFF                   call sprintf
47 0000015E A1[00000000]                   mov eax,[max]
48 00000163 E81EFFFFFF                   call iprintf
49 00000168 E86EFFFFFF                   call quit

```

Рис. 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

5. Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу

для варианта 15 - 32, 6, 54



```
call sprint
mov ecx,C
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

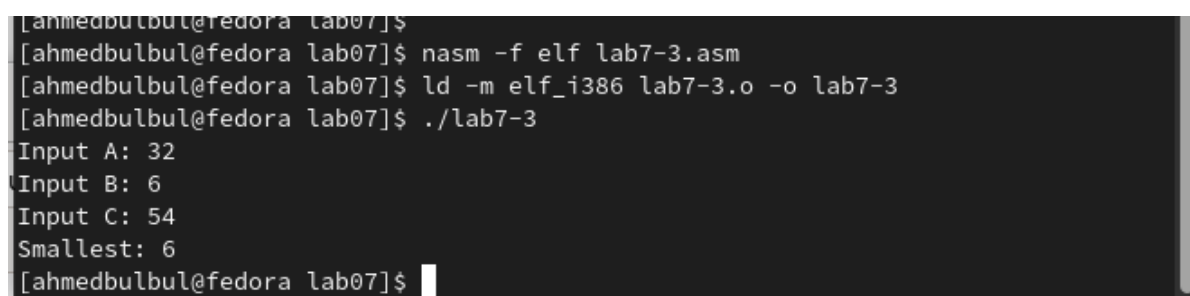
check_C:
cmp ecx, [C]
jnl finish
mov ecx,[C]
mov [min],ecx

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit
```

Рис. 2.12: Программа lab7-3.asm



```
[ahmedbulbul@fedora lab07]$
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-3.asm
[ahmedbulbul@fedora lab07]$ ld -m elf_i386 lab7-3.o -o lab7-3
[ahmedbulbul@fedora lab07]$ ./lab7-3
Input A: 32
Input B: 6
Input C: 54
Smallest: 6
[ahmedbulbul@fedora lab07]$
```

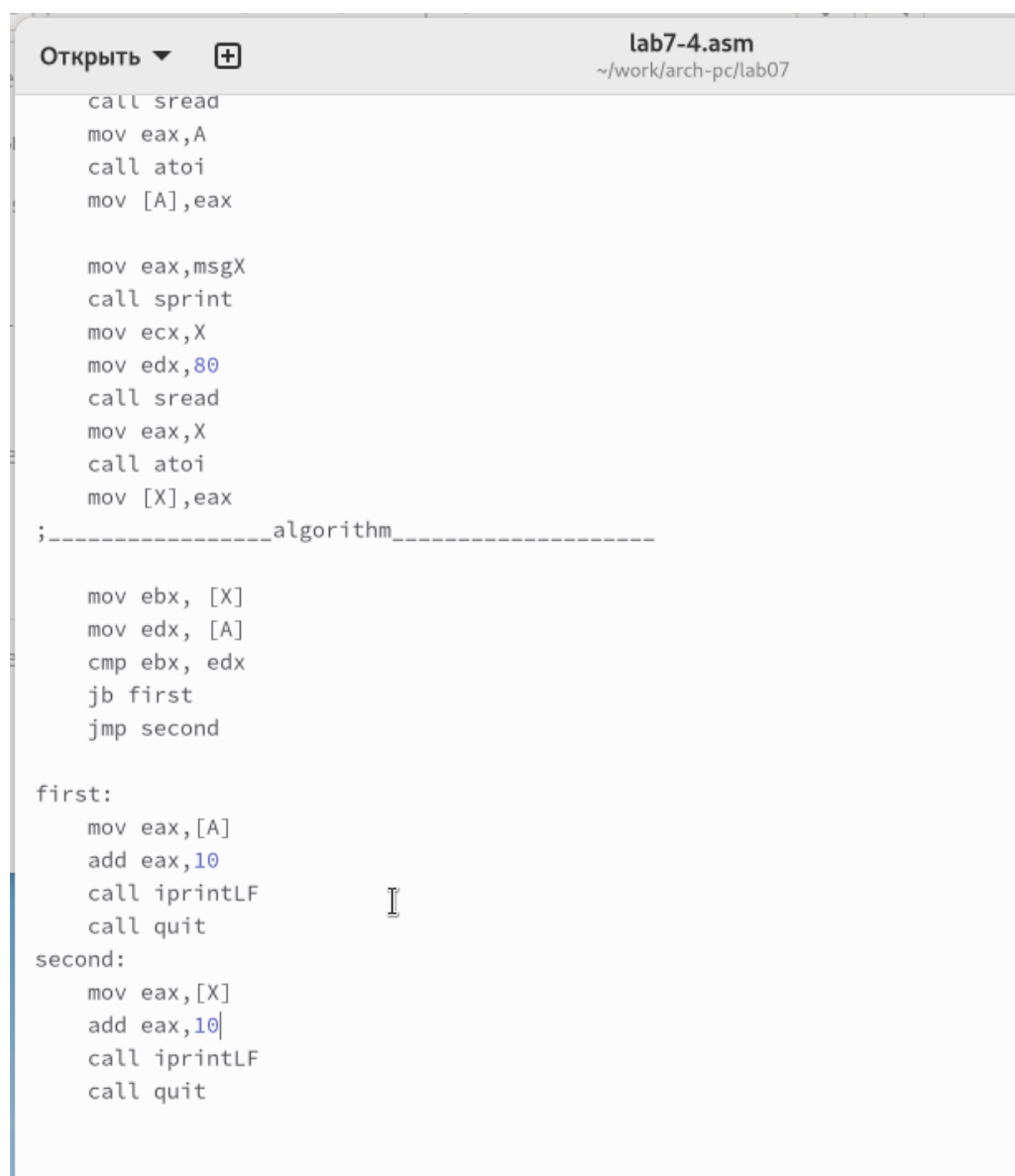
Рис. 2.13: Запуск программы lab7-3.asm

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 7.6 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 7.6.

для варианта 15

$$\begin{cases} a + 10, x < a \\ x + 10, x \geq a \end{cases}$$



```
lab7-4.asm
~/.work/arch-pc/lab07

call sread
mov eax,A
call atoi
mov [A],eax

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
jb first
jmp second

first:
mov eax,[A]
add eax,10
call iprintLF
call quit

second:
mov eax,[X]
add eax,10
call iprintLF
call quit
```

Рис. 2.14: Программа lab7-4.asm

```
[ahmedbulbul@fedora lab07]$ nasm -f elf lab7-4.asm
[ahmedbulbul@fedora lab07]$ ld -m elf_i386 lab7-4.o -o lab7-4
[ahmedbulbul@fedora lab07]$ ./lab7-4
Input A: 3
Input X: 2
13
[ahmedbulbul@fedora lab07]$ ./lab7-4
Input A: 2
Input X: 4
14
[ahmedbulbul@fedora lab07]$
```

Рис. 2.15: Запуск программы lab7-4.asm

3 Выводы

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