

Отчёта по лабораторной работе №7

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

Жозе Рамос Домингуш

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файлы листинга	10
3.3	Задание для самостоятельной работы	12
4	Выводы	16

Список иллюстраций

3.1	Создаем каталог с помощью команды <code>mkdir</code> и файл с помощью команды <code>touch</code>	6
3.2	Заполняем файл	6
3.3	Запускаем файл и смотрим на его работу	7
3.4	Изменяем файл	7
3.5	Запускаем файл и смотрим на его работу	7
3.6	Редактируем файл	8
3.7	Проверяем, сошелся ли наш вывод с данным в условии выводом .	8
3.8	Создаем файл командой <code>touch</code>	8
3.9	Заполняем файл	9
3.10	Смотрим на работу программ	9
3.11	Создаем файл листинга	10
3.12	Изучаем файл	10
3.13	Удаляем операндум из файла	11
3.14	Транслируем файл	11
3.15	Изучаем файл с ошибкой	12
3.16	Создаем файл командой <code>touch</code>	12
3.17	Пишем программу	13
3.18	Смотрим на работу программы(всё верно)	13
3.19	Создаем файл командой <code>touch</code>	14
3.20	Пишем программу	14
3.21	Проверяем работу программы	14
3.22	Проверяем работу программы	15

1 Цель работы

Освоить условного и безусловного перехода. Ознакомиться с назначением и структурой файла листинга.

2 Задание

Написать программы для решения системы выражений.

3 Выполнение лабораторной работы

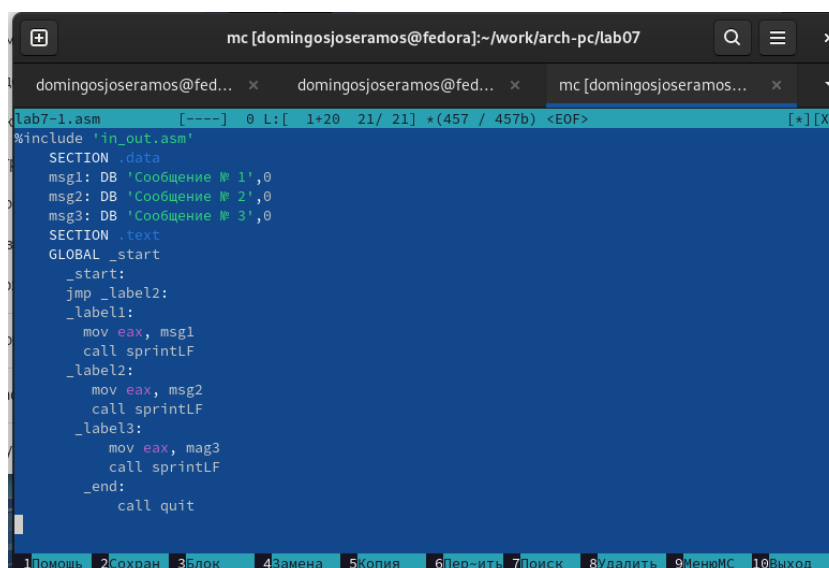
3.1 Реализация переходов в NASM

Создаем каталог для программ ЛБ7, и в нем создаем файл (рис. fig. 3.1).

```
domingosjoseramos@fedora:~$ mkdir ~/work/arch-pc/lab07
domingosjoseramos@fedora:~$ cd ~/work/arch-pc/lab07
domingosjoseramos@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.1: Создаем каталог с помощью команды `mkdir` и файл с помощью команды `touch`

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1 (рис. fig. 3.2).

The image shows a screenshot of the Midnight Commander file manager. The title bar indicates the current directory is ~/work/arch-pc/lab07. The main window displays the contents of the file lab7-1.asm. The code is written in NASM syntax and includes comments in Russian. It defines three data messages, sets up a text section, and implements a loop that prints each message and then jumps back to the start of the loop. The status bar at the bottom shows various menu options like 'Помощь', 'Сохран', 'Блок', etc.

```
lab7-1.asm  [----]  0 L: [ 1+20 21/ 21] *(457 / 457b) <EOF>  [*] [X]
%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 sprintf
_label2:
mov eax, msg2
call sprintf
_label3:
mov eax, msg3
call sprintf
_end:
call quit
```

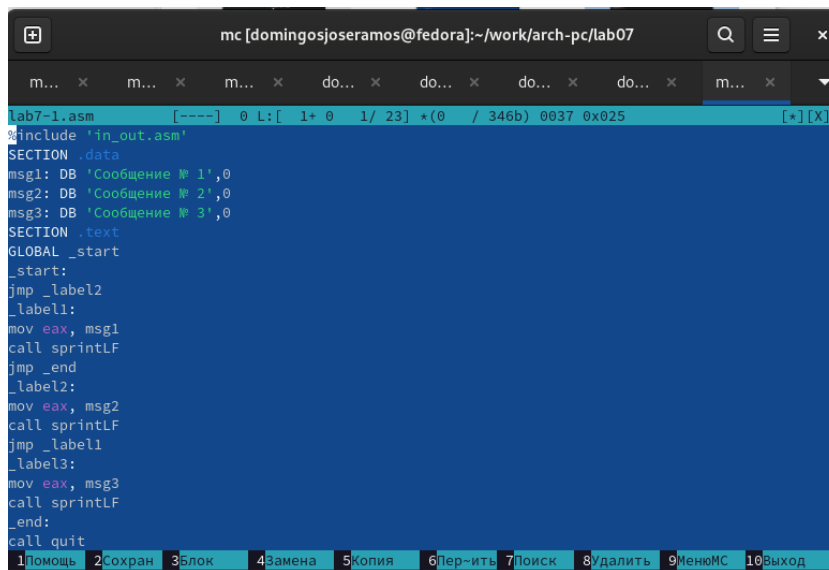
Рис. 3.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.3: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2 (рис. fig. 3.4).



```
lab7-1.asm [----] 0 L: [ 1+ 0 1/ 23] *(0 / 346b) 0037 0x025 [*][X]
%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 sprintfLF
jmp _end
_label2:
mov eax, msg2
call sprintfLF
jmp _label1
_label3:
mov eax, msg3
call sprintfLF
_end:
call quit
```

Рис. 3.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.5).

```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.5: Запускаем файл и смотрим на его работу

Снова открываем файл для редактирования и изменяем его, чтобы произошел данный вывод (рис. fig. 3.6).

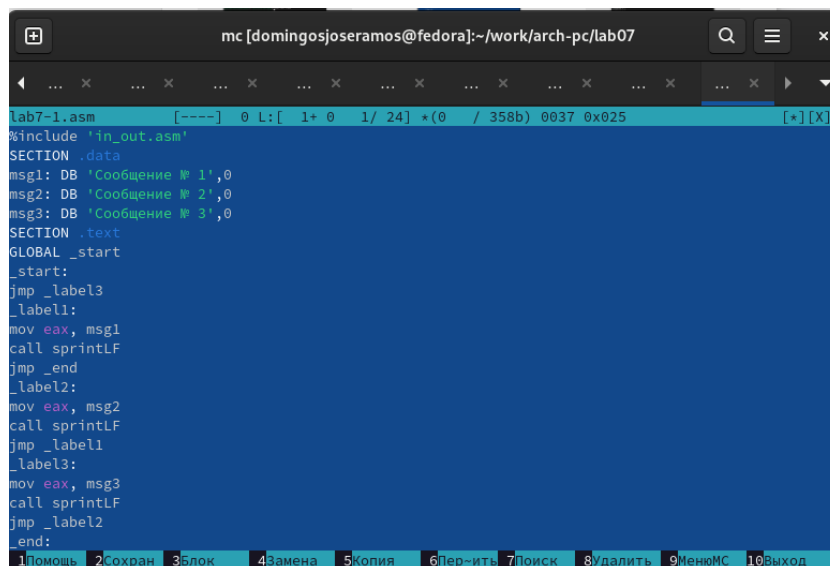


Рис. 3.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. fig. 3.7).

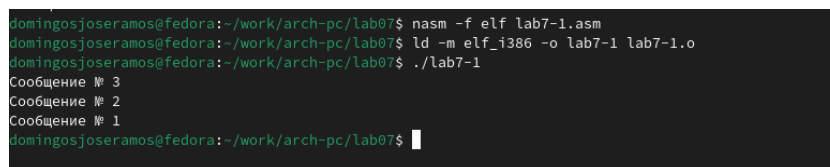


Рис. 3.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

Создаем новый файл (рис. fig. 3.8).

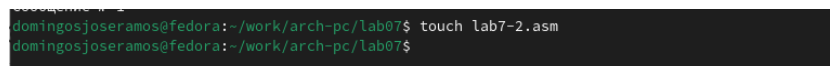
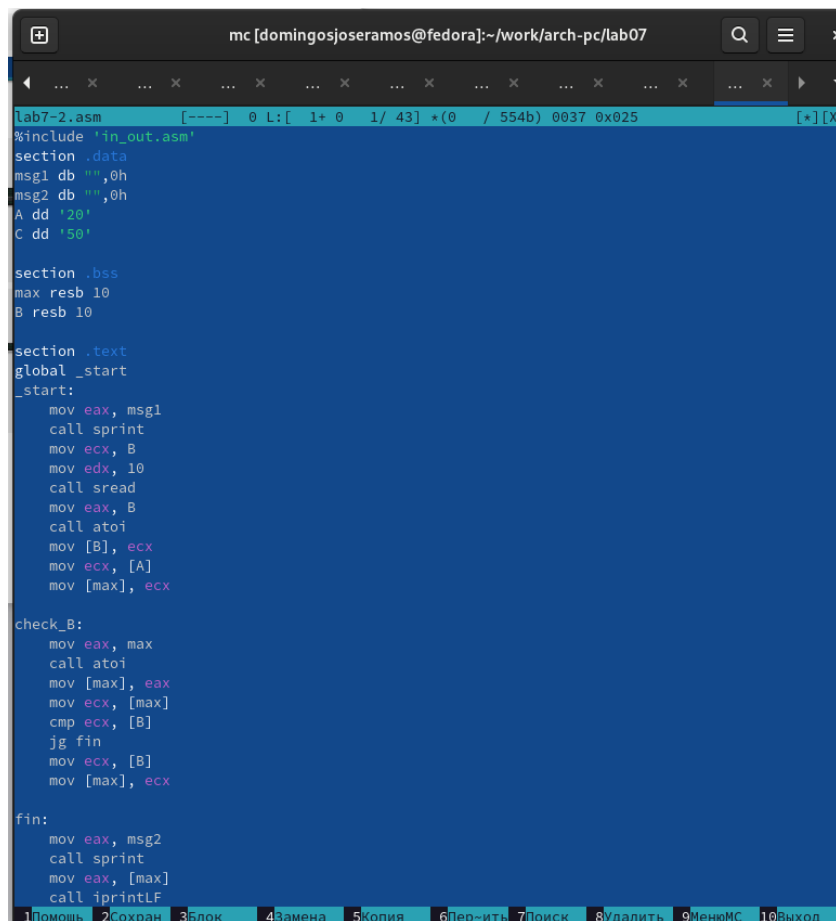


Рис. 3.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3 (рис. fig. 3.9).



```
lab7-2.asm  [----]  0 L: [ 1+ 0 1/ 43] *(0 / 554b) 0037 0x025  [*] [X]
#include 'in_out.asm'
section .data
msg1 db "",0h
msg2 db "",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], ecx
    mov ecx, [A]
    mov [max], ecx

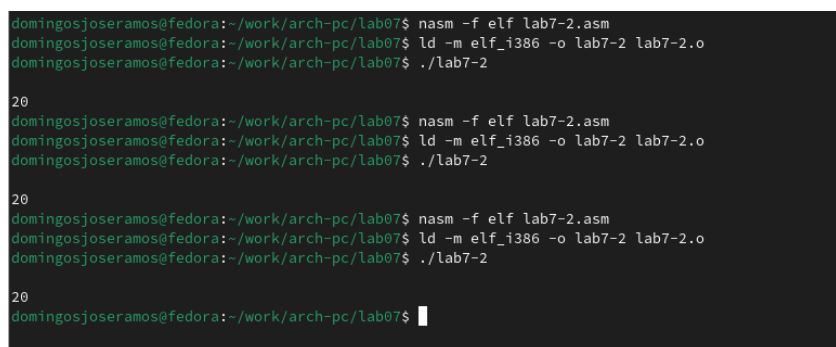
check_B:
    mov eax, max
    call atoi
    mov [max], eax
    mov ecx, [max]
    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx

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

1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 3.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, вводя разные значения В (рис. fig. 3.10).



```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-2

20
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-2

20
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-2

20
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.10: Смотрим на работу программ

3.2 Изучение структуры файлы листинга

Создаем файл листинга для программы lab7-2.asm (рис. fig. 3.11).

```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.11: Создаем файл листинга

Открываем файл листинга с помощью команды mscedit и изучаем его (рис. fig. 3.12).

```
lab7-2.lst  [----] 0 L: [ 1+ 0 1/213] *(0 /12808b) 0032 0x020 [*)(X)
1          %include 'in_out.asm'
2          <1> ;----- slen -----
3          <1> ; функция вычисления длины сообщения
4          <1> slen:-----
5          00000000 53          <1> push ebx
6          00000001 89C3        <1> mov ebx, eax
7          <1> .....
8          00000003 803800      <1> nextchar:-----
9          00000006 7403        <1> cmp byte [eax], 0
10         00000008 40          <1> jz finished
11         00000009 EBF8        <1> inc eax
12         <1> .....
13         <1> jmp nextchar
14         <1> .....
15         0000000B 29D8        <1> finished:
16         0000000D 5B          <1> sub eax, ebx
17         0000000E C3          <1> pop ebx
18         <1> .....
19         <1> .....
20         <1> ;----- sprint -----
21         <1> ; функция печати сообщения
22         <1> ; входные данные: mov eax, <message>
23         <1> sprint:
24         0000000F 52          <1> push edx
25         00000010 51          <1> push ecx
26         00000011 53          <1> push ebx
27         00000012 50          <1> push eax
28         00000013 E8E8FFFFFF <1> call slen
29         <1> .....
30         00000018 89C2        <1> mov edx, eax
31         0000001A 58          <1> pop eax
32         <1> .....
33         0000001B 89C1        <1> mov ecx, eax
34         0000001D BB01000000 <1> mov ebx, 1
35         00000022 B804000000 <1> mov eax, 4
36         00000027 CD80        <1> int 80h
37         <1> .....
38         00000029 5B          <1> pop ebx
39         0000002A 59          <1> pop ecx
40         0000002B 5A          <1> pop edx
```

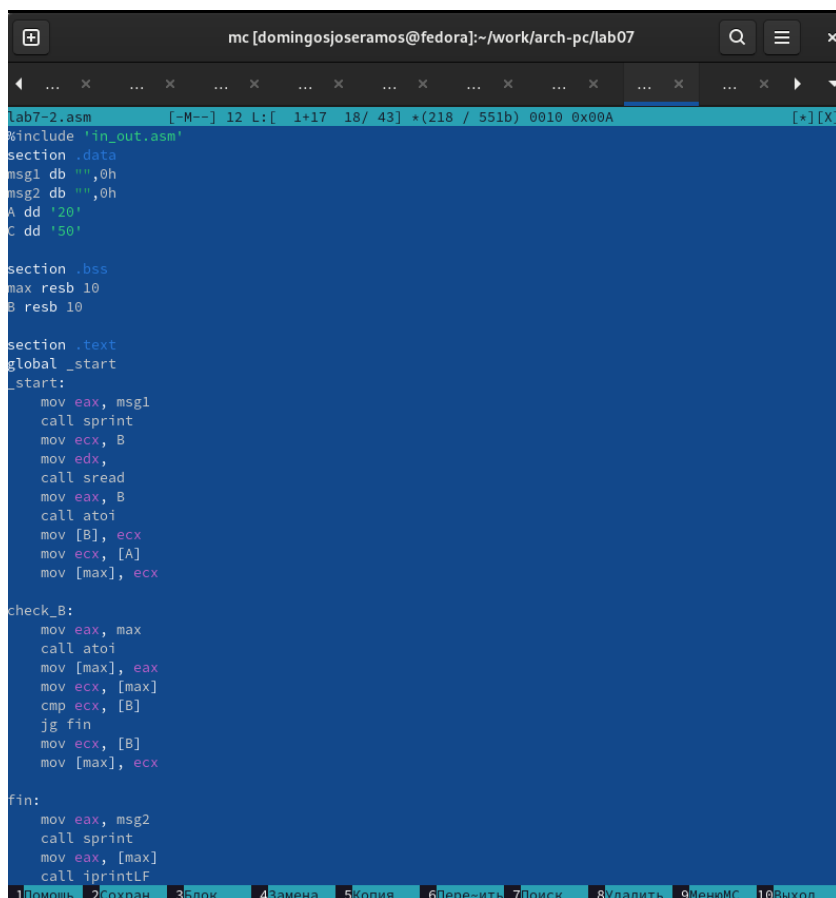
Рис. 3.12: Изучаем файл

Строка 33: 0000001D-адрес в сегменте кода, BB01000000-машинный код, mov ebx,1-присвоение переменной ebx значения 1.

Строка 34: 00000022-адрес в сегменте кода, B804000000-машинный код, mov eax,4-присвоение переменной eax значения 4.

Строка 35 00000027-адрес в сегменте кода, CD80-машинный код, int 80h-вызов ядра.

Открываем файл и удаляем один операндум (рис. fig. 3.13).



```
lab7-2.asm [-M--] 12 L: [ 1+17 18/ 43] *(218 / 551b) 0010 0x00A [*] [X]
%include 'in_out.asm'
section .data
msg1 db "",0h
msg2 db "",0h
A dd '20'
C dd '50'

section .bss
max resb 10
B resb 10

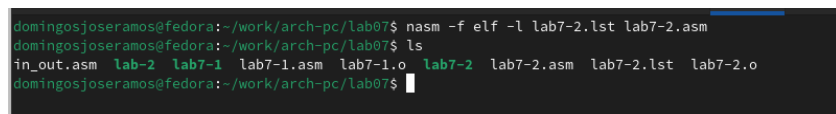
section .text
global _start
_start:
    mov eax, msg1
    call sprint
    mov ecx, B
    mov edx,
    call sread
    mov eax, B
    call atoi
    mov [B], ecx
    mov ecx, [A]
    mov [max], ecx

check_B:
    mov eax, max
    call atoi
    mov [max], eax
    mov ecx, [max]
    cmp ecx, [B]
    jg fin
    mov ecx, [B]
    mov [max], ecx

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

Рис. 3.13: Удаляем операндум из файла

Транслируем с получением файла листинга (рис. fig. 3.14).



```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab-2 lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst lab7-2.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.14: Транслируем файл

При трансляции файла, выдается ошибка, но создаются исполнительный файл lab7-2 и lab7-2.lst

Снова открываем файл листинга и изучаем его (рис. fig. 3.15).

```

mc [domingosjoseramos@fedora:~/work/arch-pc/lab07]
lab7-2.lst [----] 0 L: [ 97+ 0 97/213] *(5959/12808b) 0032 0x020 [*] [X]
96 00000076 E894FFFFFF <1> call sprint...
97 0000007B 58 <1> pop eax...
98 0000007C 83F900 <1> cmp ecx, 0...
99 0000007F 75F2 <1> jnz printLoop..
100 <1>
101 00000081 5E <1> pop esi...
102 00000082 5A <1> pop edx...
103 00000083 59 <1> pop ecx...
104 00000084 58 <1> pop eax...
105 00000085 C3 <1> ret
106 <1>
107 <1>
108 <1> ;----- iprintLF -----
109 <1> ; Функция вывода на экран чисел в формате ASCII
110 <1> ; входные данные: mov eax,<int>
111 <1> iprintLF:
112 00000086 E8C9FFFFFF <1> call iprint...
113 <1>
114 0000008B 50 <1> push eax...
115 0000008C B80A000000 <1> mov eax, 0Ah...
116 00000091 50 <1> push eax...
117 00000092 89E0 <1> mov eax, esp...
118 00000094 E876FFFFFF <1> call sprint...
119 00000099 58 <1> pop eax...
120 0000009A 58 <1> pop eax...
121 0000009B C3 <1> ret
122 <1>
123 <1> ;----- atoi -----
124 <1> ; Функция преобразования ascii-код символа в целое число
125 <1> ; входные данные: mov eax,<int>
126 <1> atoi:
127 0000009C 53 <1> push ebx...
128 0000009D 51 <1> push ecx...
129 0000009E 52 <1> push edx...
130 0000009F 56 <1> push esi...
131 000000A0 89C6 <1> mov esi, eax...
132 000000A2 B800000000 <1> mov eax, 0...
133 000000A7 B900000000 <1> mov ecx, 0...
134 <1>
135 <1> .multiplyLoop:
1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Пере-ить 7 Поиск 8 Удалить 9 МенюМС 10 Выход

```

Рис. 3.15: Изучаем файл с ошибкой

3.3 Задание для самостоятельной работы

ВАРИАНТ-13

1. 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 № 7. Создайте исполняемый файл и проверьте его работу.

Создаем новый файл (рис. fig. 3.16).

```

domingosjoseramos@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$

```

Рис. 3.16: Создаем файл командой touch

Открываем его и пишем программу, которая выберет наименьшее число из трех(2 числа уже в программе, 3е вводится из консоли) (рис. fig. 3.17).

```

lab7-3.asm  [----]  0 L: 1+ 6 7/ 48] *(126 / 674b) 0067 0x043  [*] [X]
#include "in_out.asm"

section .data
msg1 db "Введите B: ",0h
msg2 db "Наименьшее число: ",0h
A dd '95'
C dd '61'

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]
    jl check_B
    mov ecx, [C]
    mov [min], ecx

check_B:
    mov eax, min
    call atoi
    mov [min], eax
    mov ecx, [min]
    cmp ecx, [B]
    jl fin
    mov ecx, [B]
    mov [min], ecx
  
```

Рис. 3.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. fig. 3.18).

```

domingosjoseramos@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 2
Наименьшее число: 2
domingosjoseramos@fedora:~/work/arch-pc/lab07$
  
```

Рис. 3.18: Смотрим на работу программы(всё верно)

2. 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

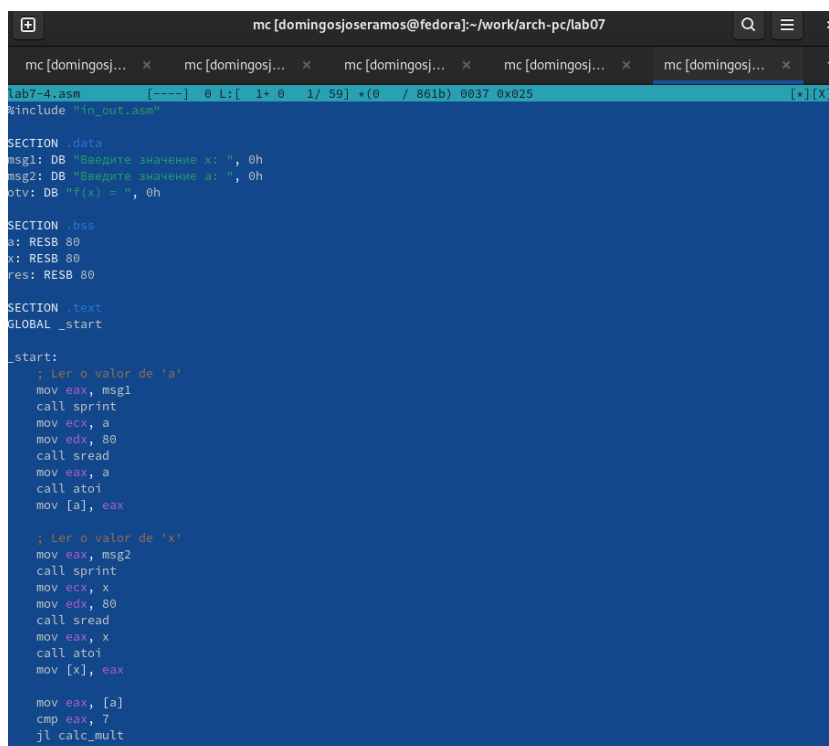
соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Создаем новый файл (рис. fig. 3.19).

```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
domingosjoseramos@fedora:~/work/arch-pc/lab07$
```

Рис. 3.19: Создаем файл командой touch

Открываем его и пишем программу, которая решит систему уравнений, при данных, введенных в консоль (рис. fig. 3.20).



```
mc [domingosj... x mc [domingosj... x mc [domingosj... x mc [domingosj... x mc [domingosj... x
lab7-4.asm [----] 0 L:[ 1+ 0 1/ 59] *(0 / 861b) 0037 0x025 [*][X]
#include "in_out.asm"

SECTION .data
msg1: DB "Введите значение x: ", 0h
msg2: DB "Введите значение a: ", 0h
ptv: DB "f(x) = ", 0h

SECTION .bss
a: RESB 80
x: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start

_start:
; Ler o valor de 'a'
mov eax, msg1
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov [a], eax

; Ler o valor de 'x'
mov eax, msg2
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov [x], eax

mov eax, [a]
cmp eax, 7
jl calc_mult
```

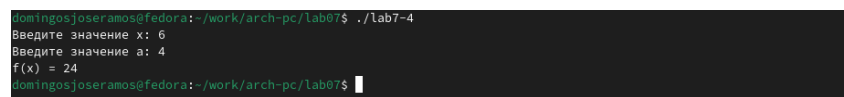
Рис. 3.20: Пишем программу

Транслируем файл и проверяем его работу при $x = 3$ и $a = 9$ (рис. fig. 3.21).

```
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение x: 3
Введите значение a: 9
f(x) = 27
domingosjoseramos@fedora:~/work/arch-pc/lab07$ ./lab7-4
```

Рис. 3.21: Проверяем работу программы

Транслируем файл и проверяем его работу при $x = 6$ и $a = 4$ (рис. fig. 3.22).



```
domingosjoseramos@fedora:~/work/arch-pc/lab67$ ./lab7-4
Введите значение x: 6
Введите значение a: 4
f(x) = 24
domingosjoseramos@fedora:~/work/arch-pc/lab67$
```

Рис. 3.22: Проверяем работу программы

4 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.