

# **Отчет по лабораторной работе №7**

**Архитектура компьютера**

Элсаиед Адел

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
2.1	1. Команды условного перехода . . . . .	6
2.2	2. Реализация переходов в NASM . . . . .	6
2.3	3. Изучение структуры файлы листинга . . . . .	6
2.4	4. Самостоятельная работа . . . . .	6
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Самостоятельная работа</b>	<b>18</b>
<b>6</b>	<b>Выводы</b>	<b>21</b>

## Список иллюстраций

4.1	Создание директории . . . . .	8
4.2	Создание копии файла для дальнейшей работы, редактирование файла . . . . .	9
4.3	Запуск исполняемого файла . . . . .	9
4.4	Редактирование программы . . . . .	10
4.5	Создание исполняемого файла . . . . .	10
4.6	Создание файла . . . . .	11
4.7	Вставляю текст в файл . . . . .	11
4.8	Вставляю текст в файл . . . . .	12
4.9	Запуск исполняемого файла . . . . .	12
4.10	Запуск исполняемого файла . . . . .	13
4.11	Файл листинга . . . . .	13
4.12	Файл листинга . . . . .	15
4.13	asm -f elf -l lab7-2.lst lab7-2.asm . . . . .	15
4.14	gedit lab7-2.lst . . . . .	16
4.15	. . . . .	16
4.16	. . . . .	17
5.1	Создание запуск файла . . . . .	18
5.2	Редактирование файла . . . . .	19
5.3	Запуск исполняемого файла . . . . .	19
5.4	создание файла . . . . .	20
5.5	ввод программы в файл . . . . .	20

## Список таблиц

# 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**

**2.1 1. Команды условного перехода**

**2.2 2. Реализация переходов в NASM**

**2.3 3. Изучение структуры файлы листинга**

**2.4 4. Самостоятельная работа**

### 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 программы без каких-либо условий.

## 4 Выполнение лабораторной работы

1

С помощью утилиты `mkdir` создаю директорию `lab07`, перехожу в нее и создаю файл для работы. (рис. [4.1]).



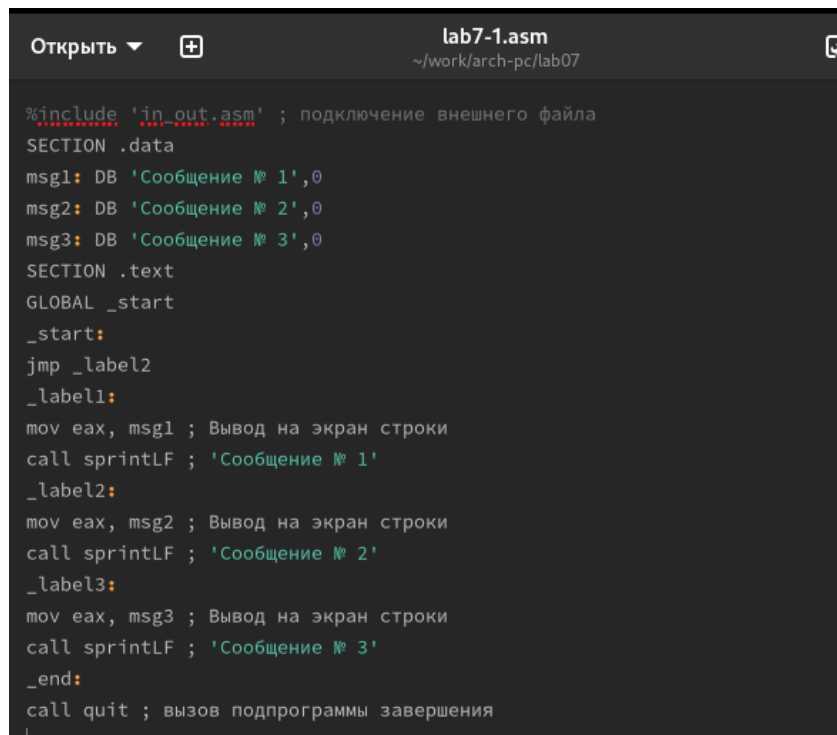
```
aaehsaied@fedora:~/work/arch-pc/lab07
[aaehsaied@fedora ~]$ mkdir ~/work/arch-pc/lab07
[aaehsaied@fedora ~]$ cd ~/work/arch-pc/lab07
[aaehsaied@fedora lab07]$ touch lab7-1.asm
[aaehsaied@fedora lab07]$
```

Рис. 4.1: Создание директории

2

Копирую в текущий каталог файл `in_out.asm` из загрузок, т.к. он будет использоваться в других программах. Открываю созданный файл `lab7-1.asm`, вставляю в него программу реализации безусловных переходов(рис. [fig002?]).





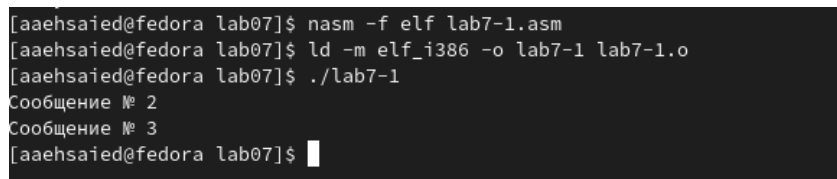
```
Открыть + lab7-1.asm
~/work/arch-pc/lab07

%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. 4.2: Создание копии файла для дальнейшей работы, редактирование файла

### 3

Создаю исполняемый файл программы и запускаю его (рис. [4.3]). Инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`.

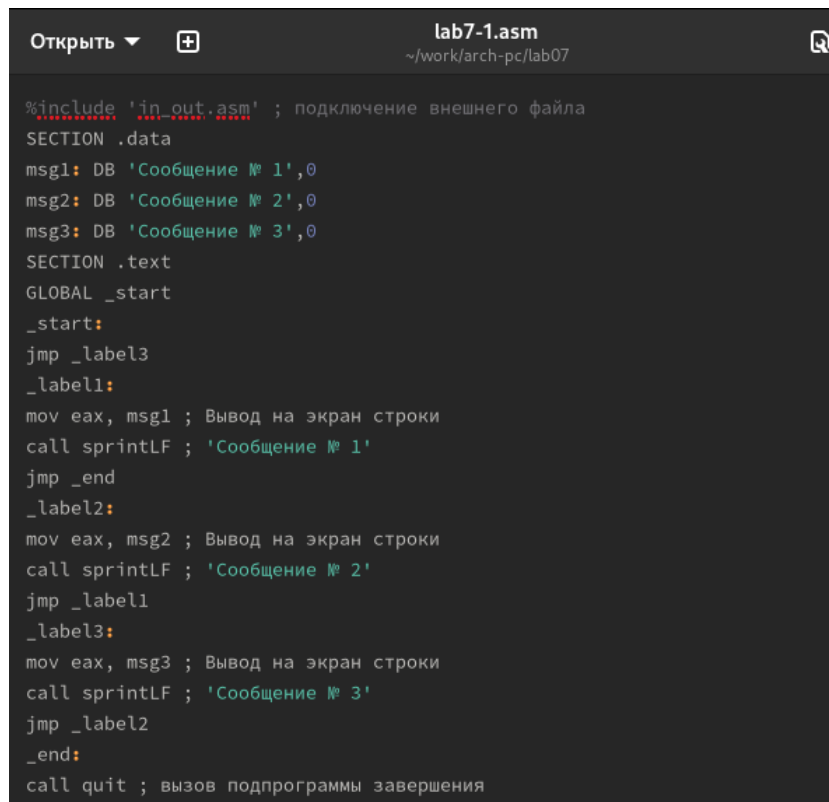


```
[aaehsaied@fedora lab07]$ nasm -f elf lab7-1.asm
[aaehsaied@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[aaehsaied@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[aaehsaied@fedora lab07]$
```

Рис. 4.3: Запуск исполняемого файла

### 4

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [4.4]).



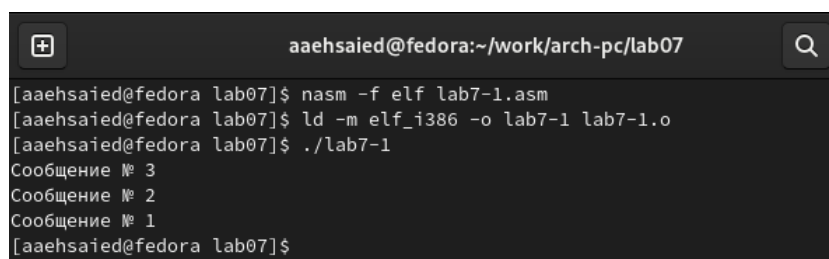
```
Открыть + lab7-1.asm ~/work/arch-pc/lab07

%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. 4.4: Редактирование программы

## 5

Создаю исполняемый файл и проверяю работу программы (рис. [4.5]). Программа отработала верно.



```
aaehsaied@fedora:~/work/arch-pc/lab07

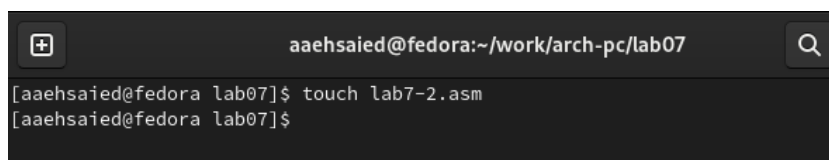
[aaehsaied@fedora lab07]$ nasm -f elf lab7-1.asm
[aaehsaied@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[aaehsaied@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[aaehsaied@fedora lab07]$
```

Рис. 4.5: Создание исполняемого файла

## 6

Создаю новый файл lab7-2.asm для программы с условным оператором. (рис.

[4.6]).

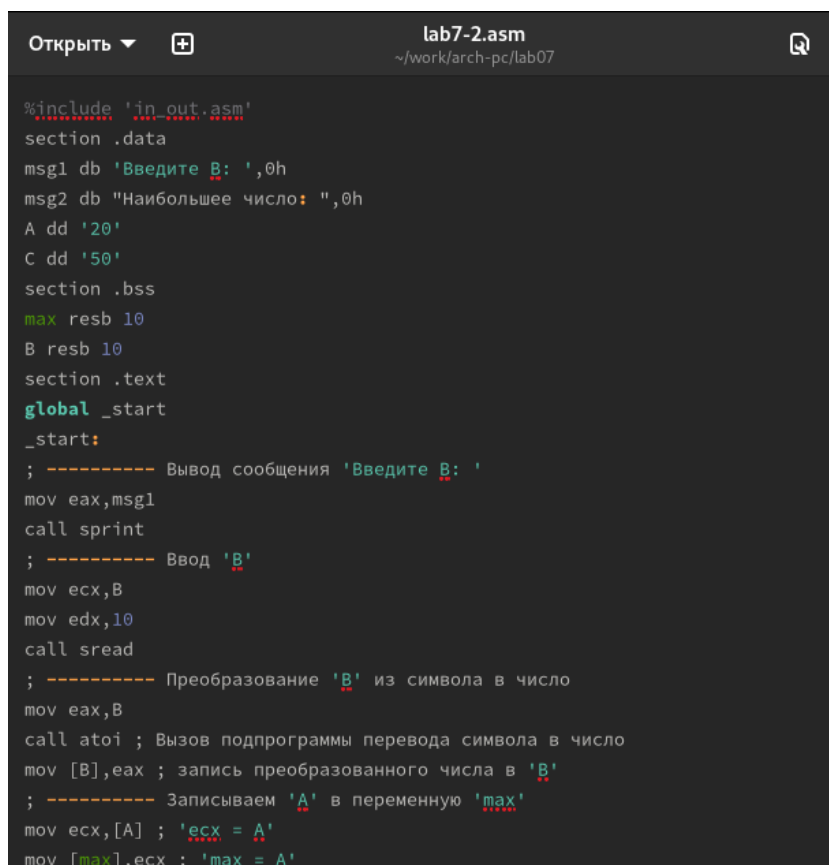


```
aaehsaied@fedora:~/work/arch-pc/lab07
[aaehsaied@fedora lab07]$ touch lab7-2.asm
[aaehsaied@fedora lab07]$
```

Рис. 4.6: Создание файла

7

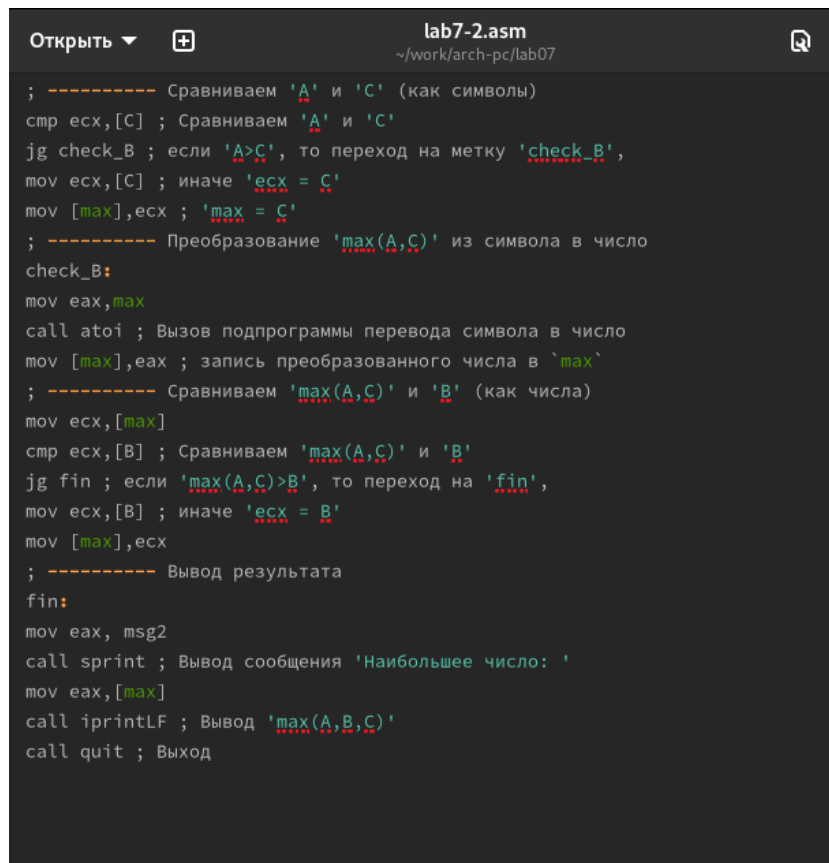
Вставляю программу, которая определяет и выводит на экран наибольшее число (рис.[4.8]).



```
Открыть ▾  lab7-2.asm
~/work/arch-pc/lab07

%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'
```

Рис. 4.7: Вставляю текст в файл

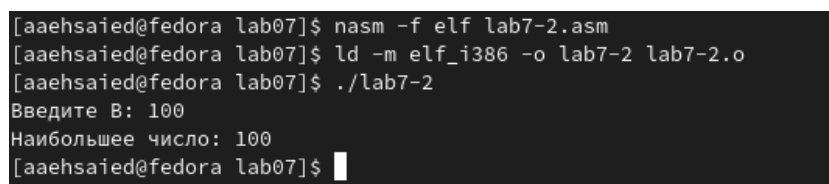


```
Открыть ▾ + lab7-2.asm ~/work/arch-pc/lab07
; ----- Сравниваем 'A' и 'C' (как символы)
ср есх,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov есх,[C] ; иначе 'есх = C'
mov [max],есх ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov еах,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],еах ; запись преобразованного числа в `max`
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov есх,[max]
ср есх,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov есх,[B] ; иначе 'есх = B'
mov [max],есх
; ----- Вывод результата
fin:
mov еах, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov еах,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 4.8: Вставляю текст в файл

## 8

Создаю и запускаю новый исполняемый файл, проверяю работу программы (рис. [4.10]).



```
[aaehsaied@fedora lab07]$ nasm -f elf lab7-2.asm
[aaehsaied@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[aaehsaied@fedora lab07]$ ./lab7-2
Введите B: 100
Наибольшее число: 100
[aaehsaied@fedora lab07]$
```

Рис. 4.9: Запуск исполняемого файла

```
[aaehsaied@fedora lab07]$ nasm -f elf lab7-2.asm
[aaehsaied@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[aaehsaied@fedora lab07]$ ./lab7-2
Введите B: 1
Наибольшее число: 50
[aaehsaied@fedora lab07]$
```

Рис. 4.10: Запуск исполняемого файла

9

Открываю файл листинга с помощью редактора mcedit. Рассмотрим 9-11 строки: (рис. [4.11]).

```
%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'
```

Рис. 4.11: Файл листинга

9 строка:

- Первые цифры [9] - это номер строки файла листинга.

- Следующие цифры [00000006] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [7403] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [jz finished] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями.

10 строка:

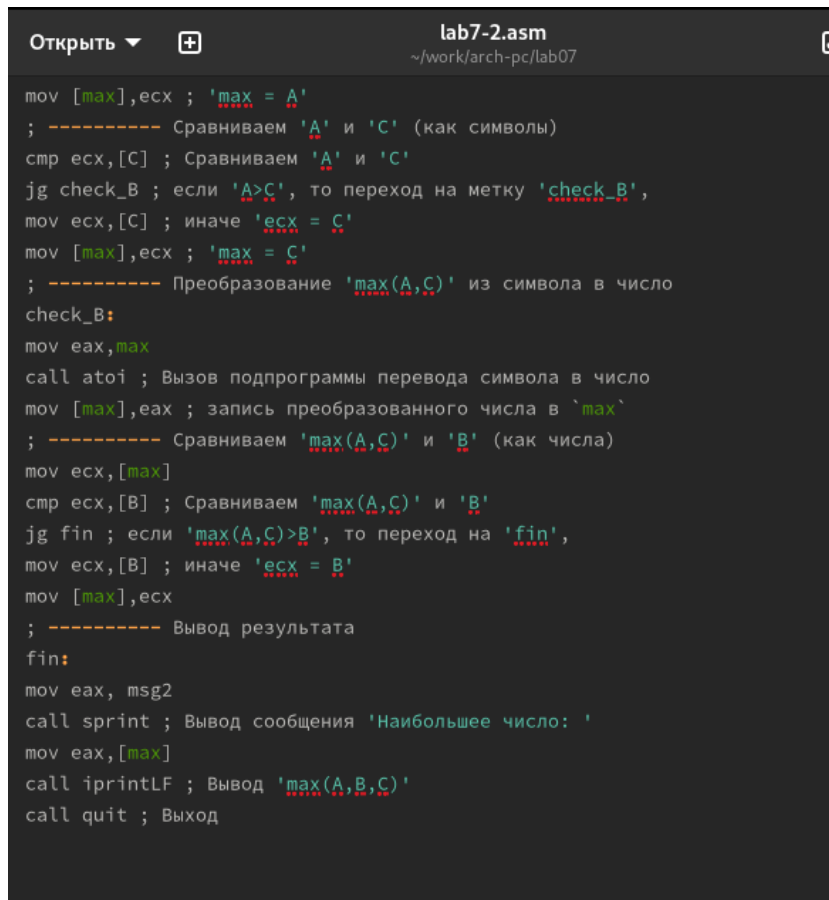
- Первые цифры [10] - это номер строки файла листинга.
- Следующие цифры [00000008] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [40] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [inc eax] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

11 строка:

- Первые цифры [11] - это номер строки файла листинга.
- Следующие цифры [00000009] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа [EBF8] - это машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатеричной последовательности, поэтому и появляются буквы латинского алфавита.
- следующее [jmp nextchar] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями

10

Открываю файл листинга с помощью редактора mcedit и замечаю, что в файле листинга появляется ошибка. (рис. [4.12]).



```
Открыть ▾ + lab7-2.asm
~/work/arch-pc/lab07

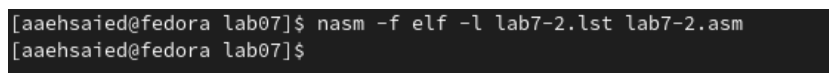
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi ; 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 sprintf ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call fprintf ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 4.12: Файл листинга

Отсюда можно сделать вывод, что, если в коде появляется ошибка, то ее описание появится в файле листинга

11

Создал файл листинга для программы из файла lab7-2.asm (рис. [4.13]).



```
[aaehsaied@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
[aaehsaied@fedora lab07]$
```

Рис. 4.13: asm -f elf -l lab7-2.lst lab7-2.asm

12

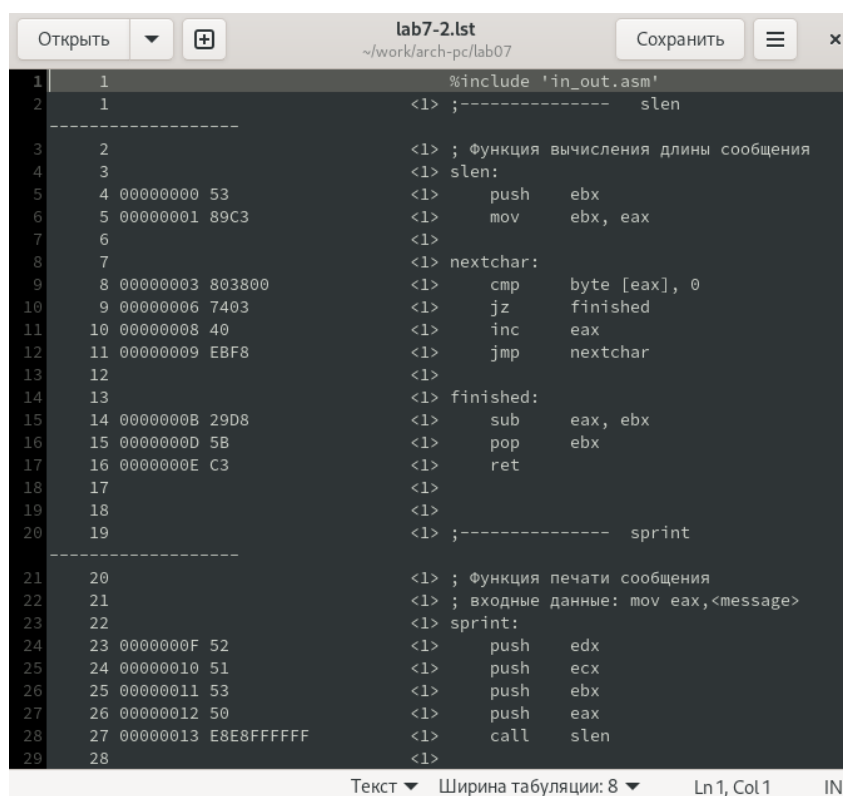
Открыл файл листинга lab7-2.lst с помощью любого текстового редактора, например gedit: {#fig:012 width=70%}

```
[aaehsaied@fedora lab07]$ gedit lab7-2.lst
```

Рис. 4.14: gedit lab7-2.lst

## 13

Открыл файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалить один операнд. Выполните трансляцию с получением файла листинга: {#fig:013 width=70%}



```
1 1 %include 'in_out.asm'
2 1 <1> ;----- slen
3 2
4 3
5 4 00000000 53 <1> ; Функция вычисления длины сообщения
6 5 00000001 89C3 <1> slen:
7 6 <1> push ebx
8 7 <1> mov ebx, eax
9 8 00000003 803800 <1> nextchar:
10 9 00000006 7403 <1> cmp byte [eax], 0
11 10 00000008 40 <1> jz finished
12 11 00000009 EBF8 <1> inc eax
13 12 <1> jmp nextchar
14 13
15 14 0000000B 29D8 <1> finished:
16 15 0000000D 5B <1> sub eax, ebx
17 16 0000000E C3 <1> pop ebx
18 17 <1> ret
19 18
20 19 <1> ;----- sprintf
21 20
22 21 <1> ; Функция печати сообщения
23 22 <1> ; входные данные: mov eax, <message>
24 23 0000000F 52 <1> sprintf:
25 24 00000010 51 <1> push edx
26 25 00000011 53 <1> push ecx
27 26 00000012 50 <1> push ebx
28 27 00000013 E8E8FFFFFF <1> push eax
29 28 <1> call slen
```

Рис. 4.15:



Открыть

+

lab7-2.lst

Сохранить

☰

✕

~/work/arch-pc/lab07

```

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

```

Текст

Ширина табуляции: 8

Ln 1, Col 1

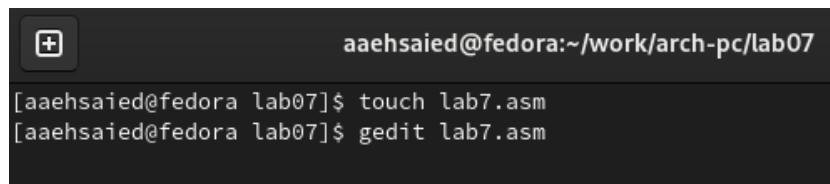
IN

Рис. 4.16:

## 5 Самостоятельная работа

1

Создаю файл lab7.asm с помощью утилиты touch и запускаю редактора gedit (рис. [5.1]).

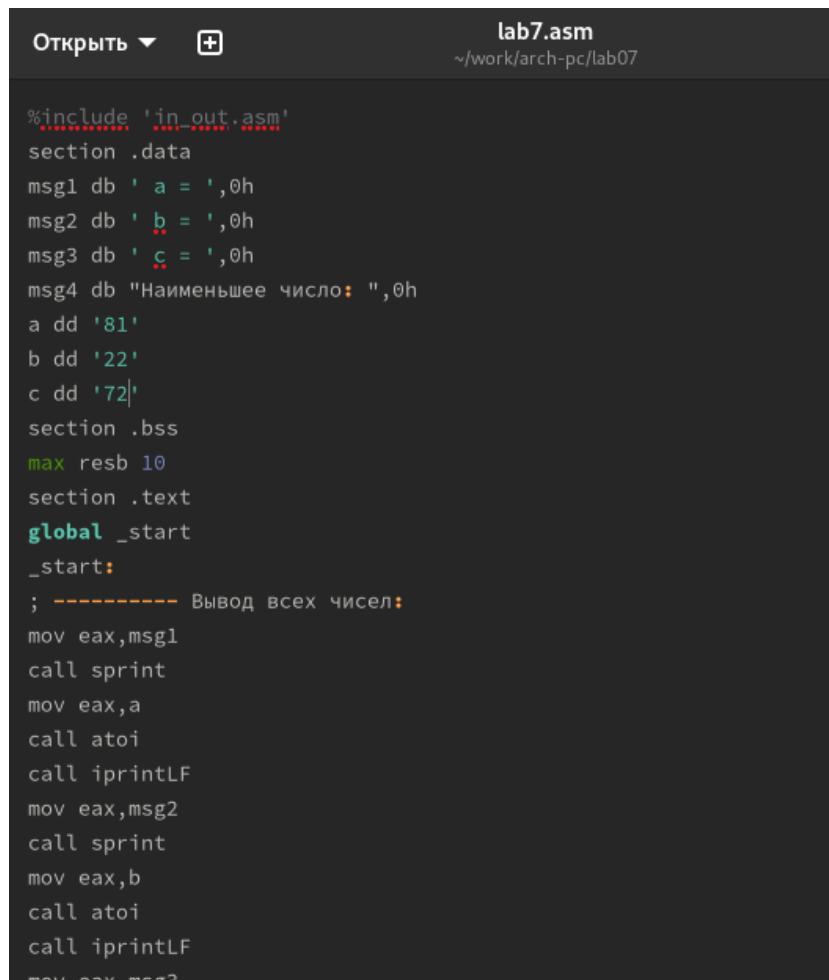
A terminal window with a dark background. The title bar shows a window icon and the text 'aaehsaied@fedora:~/work/arch-pc/lab07'. The terminal contains two lines of text: '[aaehsaied@fedora lab07]\$ touch lab7.asm' and '[aaehsaied@fedora lab07]\$ gedit lab7.asm'.

```
aaehsaied@fedora:~/work/arch-pc/lab07
[aaehsaied@fedora lab07]$ touch lab7.asm
[aaehsaied@fedora lab07]$ gedit lab7.asm
```

Рис. 5.1: Создание запуск файла

2

Ввожу в созданный файл текст программы для вычисления наименьшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы. 14 вариант (рис. [5.2]).



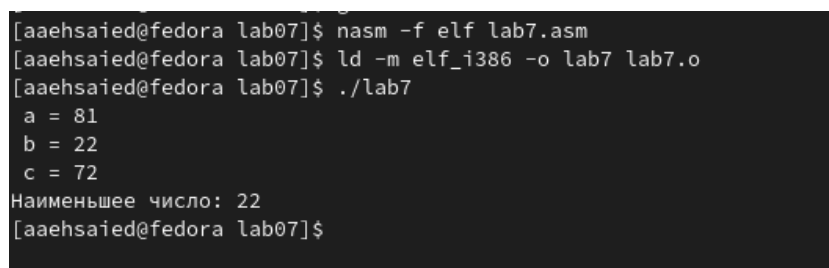
```
Открыть ▾ + lab7.asm
~/work/arch-pc/lab07

%include 'in_out.asm'
section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наименьшее число: ",0h
a dd '81'
b dd '22'
c dd '72'
section .bss
max resb 10
section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF
mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF
mov eax,msg3
```

Рис. 5.2: Редактирование файла

### 3

Создаю исполняемый файл и запускаю его (рис. [5.3]).

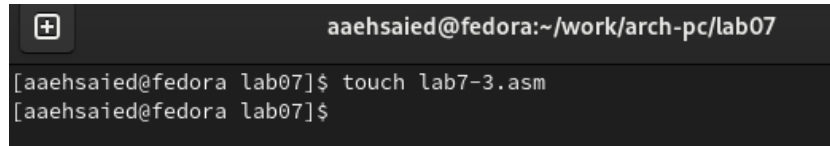


```
[aaehsaied@fedora lab07]$ nasm -f elf lab7.asm
[aaehsaied@fedora lab07]$ ld -m elf_i386 -o lab7 lab7.o
[aaehsaied@fedora lab07]$ ./lab7
a = 81
b = 22
c = 72
Наименьшее число: 22
[aaehsaied@fedora lab07]$
```

Рис. 5.3: Запуск исполняемого файла

4

Создаю новый файл lab7-3 для написания программы второго задания. (рис. [5.4]).

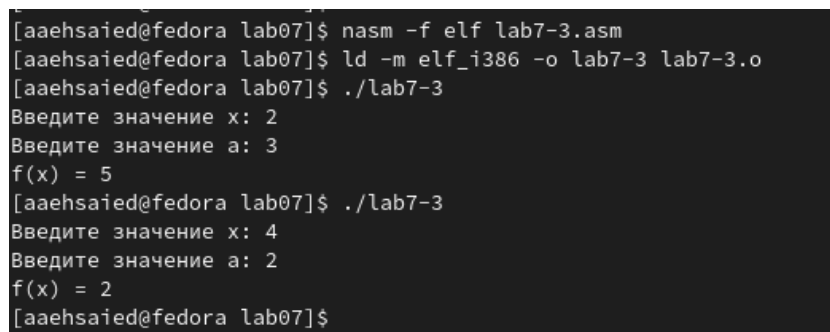
A terminal window with a dark background. The title bar shows a plus icon and the text 'aaehsaied@fedora:~/work/arch-pc/lab07'. The terminal content shows two lines of command and output: '[aaehsaied@fedora lab07]\$ touch lab7-3.asm' followed by '[aaehsaied@fedora lab07]\$' on the next line.

```
aaehsaied@fedora:~/work/arch-pc/lab07
[aaehsaied@fedora lab07]$ touch lab7-3.asm
[aaehsaied@fedora lab07]$
```

Рис. 5.4: создание файла

5

Ввожу в него программу, в которую ввожу значения 14 x и a, и которая выводит значения функции. Функцию беру из таблицы в соответствии со своим вариантом (рис. [5.5]).

A terminal window with a dark background. The title bar shows a plus icon and the text 'aaehsaied@fedora:~/work/arch-pc/lab07'. The terminal content shows several lines of command and output: '[aaehsaied@fedora lab07]\$ nasm -f elf lab7-3.asm', '[aaehsaied@fedora lab07]\$ ld -m elf\_i386 -o lab7-3 lab7-3.o', '[aaehsaied@fedora lab07]\$ ./lab7-3', followed by two prompts 'Введите значение x:' and 'Введите значение a:' with inputs '2' and '3' respectively, then 'f(x) = 5'. This is repeated with inputs '4' and '2' resulting in 'f(x) = 2'. The terminal ends with '[aaehsaied@fedora lab07]\$' on the last line.

```
aaehsaied@fedora:~/work/arch-pc/lab07
[aaehsaied@fedora lab07]$ nasm -f elf lab7-3.asm
[aaehsaied@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[aaehsaied@fedora lab07]$ ./lab7-3
Введите значение x: 2
Введите значение a: 3
f(x) = 5
[aaehsaied@fedora lab07]$ ./lab7-3
Введите значение x: 4
Введите значение a: 2
f(x) = 2
[aaehsaied@fedora lab07]$
```

Рис. 5.5: ввод программы в файл

## **6 Выводы**

При выполнении данной лабораторной работы я освоил инструкции условного и безусловного вывода и ознакомился с структурой файла листинга.