

# **Очёт по лабораторной работе № 8**

**НММБД-02-22**

Нати Франсишку Бунда

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Реализация переходов в NASM . . . . .	6
3.2	Изучение структуры файлы листинга . . . . .	11
3.3	Задание для самостоятельной работы . . . . .	13
<b>4</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

3.1	lab8-1.asm . . . . .	6
3.2	Текст программы . . . . .	7
3.3	Результат работы . . . . .	7
3.4	Использование инструкций . . . . .	8
3.5	Текст программы . . . . .	8
3.6	Инструкции jmp . . . . .	9
3.7	Исполняемый файл . . . . .	9
3.8	lab8-2.asm . . . . .	10
3.9	Текст программы . . . . .	10
3.10	Исполняемый файл . . . . .	10
3.11	Ключ -l . . . . .	11
3.12	mcedit . . . . .	11
3.13	lab8-2.lst . . . . .	12
3.14	lab8-2.asm . . . . .	13
3.15	mcedit . . . . .	13
3.16	lab8-2.lst . . . . .	13
3.17	lab8-3.asm . . . . .	14
3.18	Исполняемый файл . . . . .	14
3.19	lab8-4.asm . . . . .	15
3.20	Исполняемый файл . . . . .	15

# 1 Цель работы

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

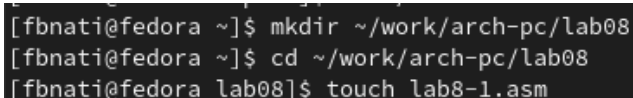
## 2 Задание

1. Реализовать переходы в NASM
2. Изучить структуру файлов листинга
3. Выполнить задание для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация переходов в NASM

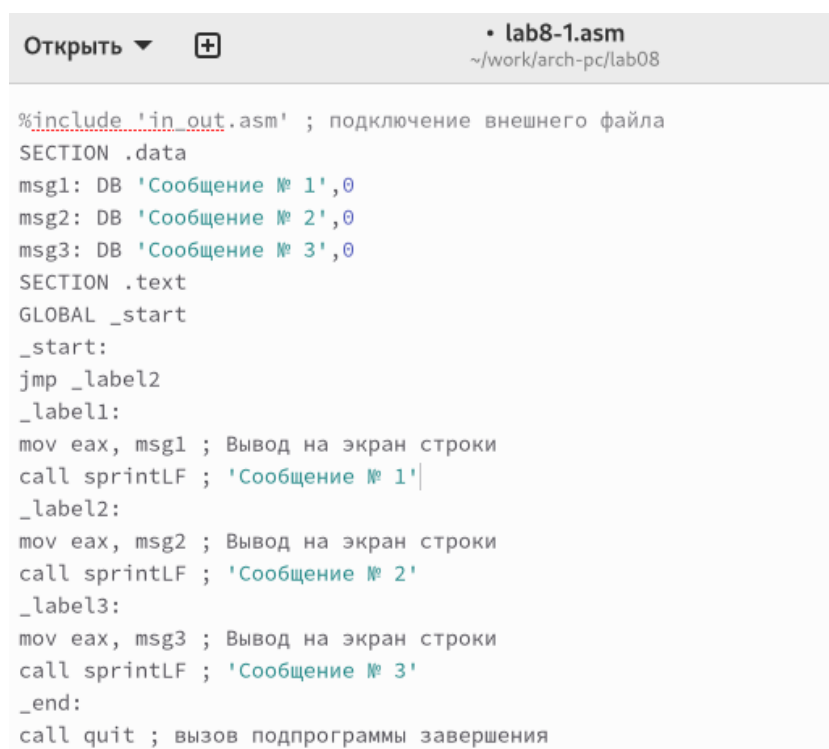
1. Создали каталог для программ лабораторной работы № 8, перешли в него и создали файл lab8-1.asm: (рис. 3.1)



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

Рис. 3.1: lab8-1.asm

2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрели пример программы с использованием инструкции `jmp`. Ввели в файл lab8-1.asm текст программы из листинга 8.1. (рис. 3.2)

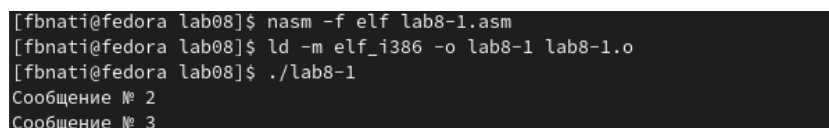


```
Открыть ▾ + • lab8-1.asm
~/work/arch-pc/lab08

%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 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. 3.2: Текст программы

Создали исполняемый файл и запустили его. Результат работы данной программы следующий: (рис. 3.3)

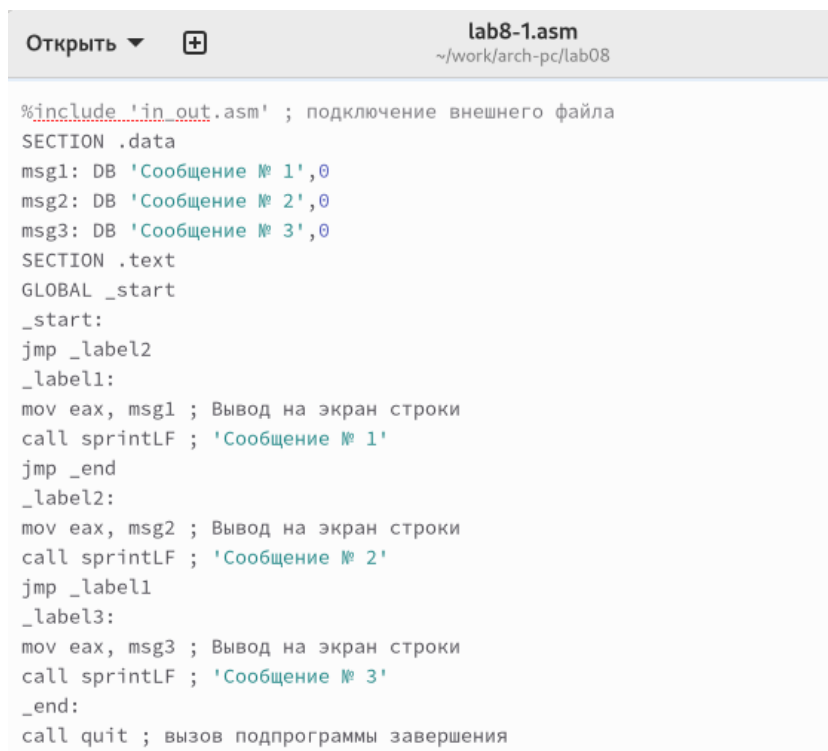


```
[fbnati@fedora lab08]$ nasm -f elf lab8-1.asm
[fbnati@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fbnati@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
```

Рис. 3.3: Результат работы

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменили программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавили инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавили инструкцию `jmp` с меткой `_end`

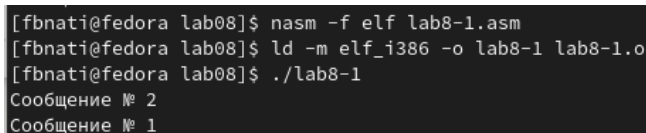
(т.е. переход к инструкции `call quit`). Изменили текст программы в соответствии с листингом 8.2 (рис. 3.4), (рис. 3.5)



```
lab8-1.asm
~/work/arch-pc/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 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. 3.4: Использование инструкций

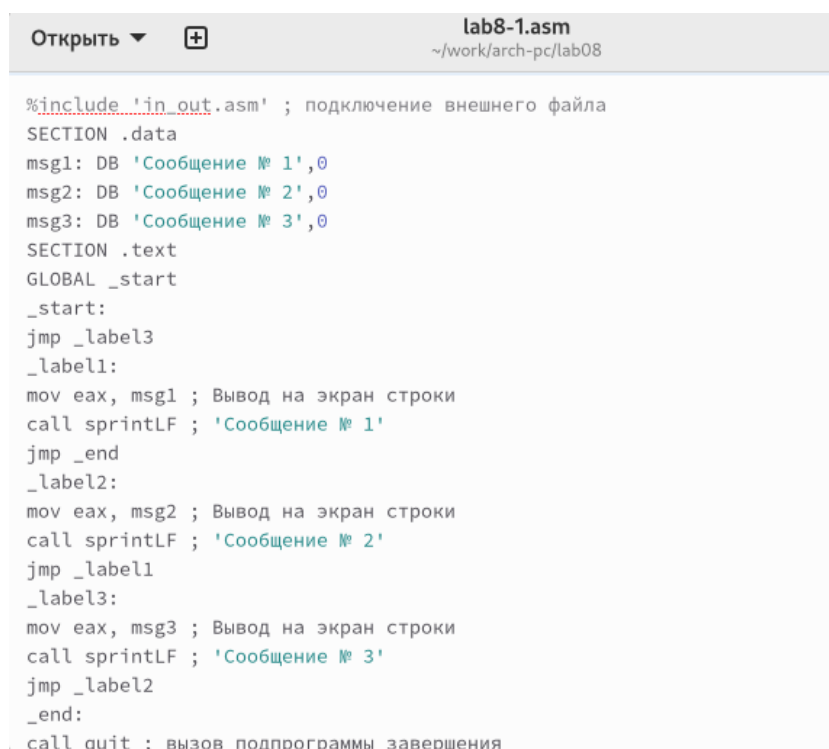


```
[fbnati@fedora lab08]$ nasm -f elf lab8-1.asm
[fbnati@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fbnati@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Текст программы

Измените текст программы добавив или изменив инструкции `jmp`. (рис. 3.6), (рис. 3.7)

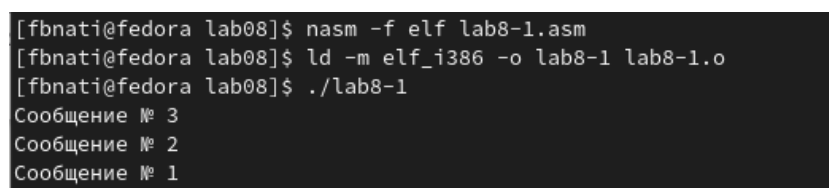




```
Открыть ▾ + lab8-1.asm
~/work/arch-pc/lab08

%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. 3.6: Инструкции jmp



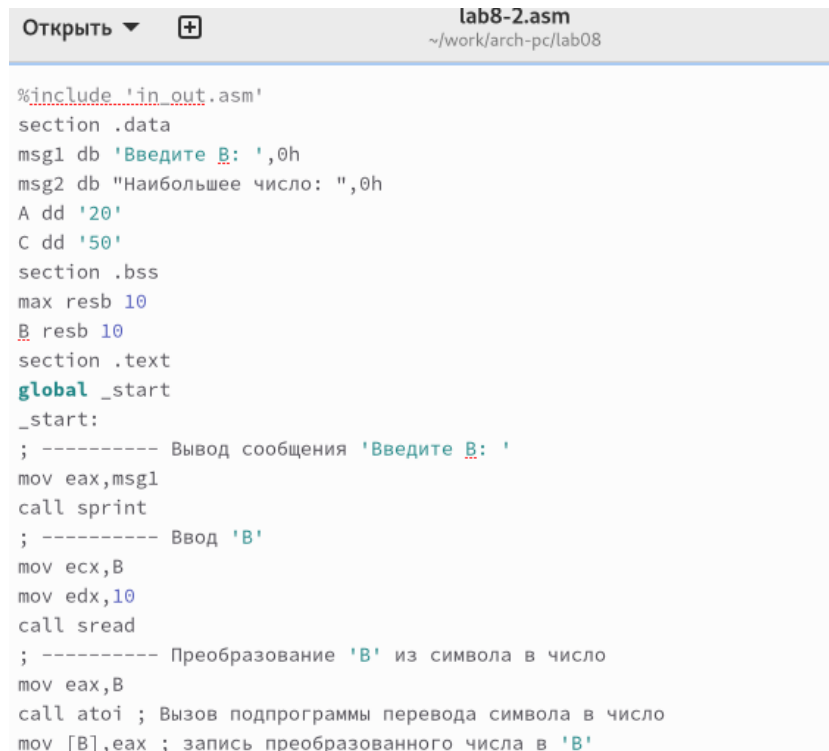
```
[fbnati@fedora lab08]$ nasm -f elf lab8-1.asm
[fbnati@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[fbnati@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 3.7: Исполняемый файл

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 рассмотрели 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. Создали файл lab8-2.asm в каталоге ~/work/arch-pc/lab08. (рис. 3.8) Внимательно изучили текст программы из листинга 8.3 и ввели в lab8-2.asm. (рис. 3.9)

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

Рис. 3.8: lab8-2.asm

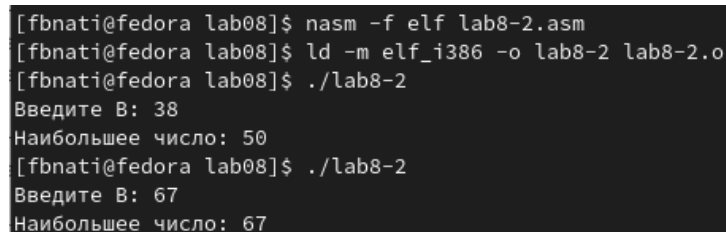


```
lab8-2.asm
~/work/arch-pc/lab08

%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'
```

Рис. 3.9: Текст программы

Создали исполняемый файл и проверили его работу для разных значений B.  
(рис. 3.10)



```
[fbnati@fedora lab08]$ nasm -f elf lab8-2.asm
[fbnati@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[fbnati@fedora lab08]$ ./lab8-2
Введите B: 38
Наибольшее число: 50
[fbnati@fedora lab08]$ ./lab8-2
Введите B: 67
Наибольшее число: 67
```

Рис. 3.10: Исполняемый файл

Обратили внимание, в данном примере переменные A и C сравниваются как символы, а переменная B и максимум из A и C как числа (для этого используется функция `atoi` преобразования символа в число). Это сделано для демонстрации

того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию atoi). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.

## 3.2 Изучение структуры файлы листинга

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. Создали файл листинга для программы из файла lab8-2.asm. (рис. 3.11)

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

Рис. 3.11: Ключ -l

Открыли файл листинга lab8-2.lst с помощью текстового редактора mcedit: (рис. 3.12), (рис. 3.13)

```
[fbnati@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 3.12: mcedit

```

fbnati@fedora:~/work/arch-pc/lab08 — mcedit lab8-2.lst
lab8-2.lst  [----]  0 L:[179+ 0 179/225] *(10993/14458b) 0032 0x020[*][X]
4 0000002E D0BBD0BE3A2000.... A dd '20'
5 00000035 32300000          C dd '50'
6 00000039 35300000          section .bss
7                                max resb 10
8 00000000 <res Ah>          B resb 10
9 0000000A <res Ah>          section .text
10                             global _start
11                             _start:
12                             ; ----- Вывод сообщения 'Введите В:
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E81DFFFFFF      call sprint
16                             ; ----- Ввод 'В'
17 000000F2 B9[0A000000]      mov ecx,B
18 000000F7 BA0A000000      mov edx,10
19 000000FC E842FFFFFF      call sread
20                             ; ----- Преобразование 'В' из симво
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      call atoi ; Вызов подпрограммы перевода
23 0000010B A3[0A000000]      mov [B],eax ; запись преобразованного чи
24                             ; ----- Записываем 'A' в переменную
25 00000110 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюMC10Выход

```

Рис. 3.13: lab8-2.lst

Внимательно ознакомились с его форматом и содержимым. Содержимое трёх строк файла листинга.

1. 45 00000154 B8[13000000] mov eax, msg2 - строка 45, адрес 00000154, B8[13000000] - машинный код, mov eax, msg2 - исходный текст программы.
2. 46 00000159 E8B1FEFFFF call sprint - строка 46, адрес 00000159, E8B1FEFFFF - машинный код, call sprint - исходный текст программы.
3. 47 0000015E A1[00000000] mov eax,[max] - строка 47, адрес 0000015E, A1[00000000] - машинный код, mov eax,[max] - исходный текст программы.

Открыли файл с программой lab8-2.asm и в инструкции mov с двумя операндами удалить один операнд. (рис. 3.14) Выполнили трансляцию с получением файла листинга: (рис. 3.15), (рис. 3.16)

```

call sprint
; ----- Ввод 'B'
mov ecx,|
mov edx,10
call sread

```

Рис. 3.14: lab8-2.asm

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

Рис. 3.15: mcedit

```

16                                     ; ----- Ввод 'B'
17                                     mov ecx,
17          *****                  error: invalid combination of opcode and operands
18 000000F2 BA0A000000                mov edx,10

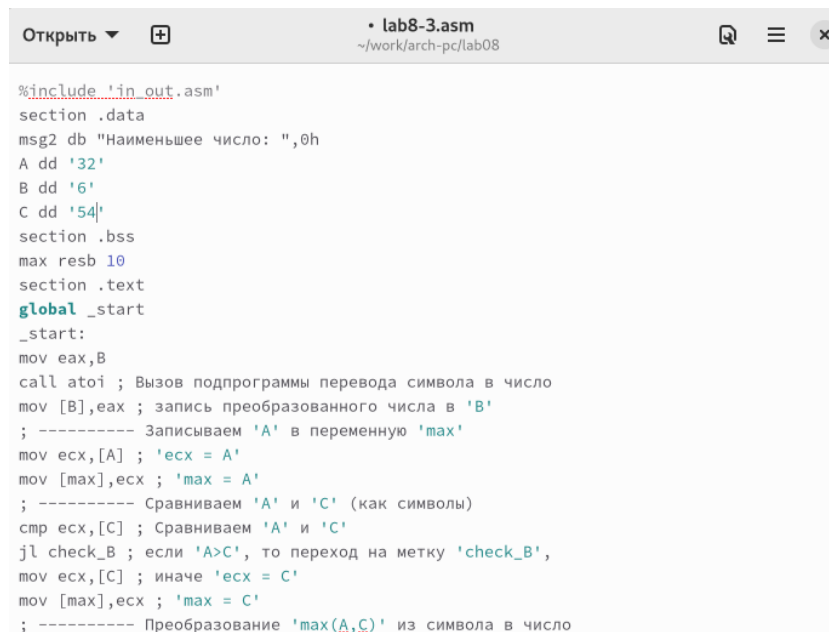
```

Рис. 3.16: lab8-2.lst

Создаётся выходной файл lst. В листинге добавляется сообщение об ошибке.

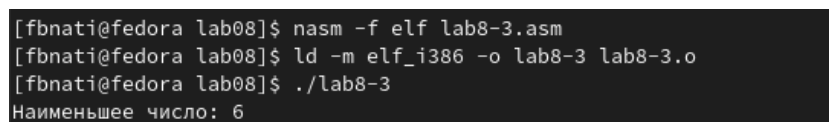
### 3.3 Задание для самостоятельной работы

1. Написали программу нахождения наименьшей из 3 целочисленных переменных a, b и c. (рис. 3.17) Значения переменных выбрали из таблицы в соответствии с 15 вариантом, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу. (рис. 3.18)



```
%include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd '32'
B dd '6'
C dd '54'
section .bss
max resb 10
section .text
global _start
_start:
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'
jle 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
```

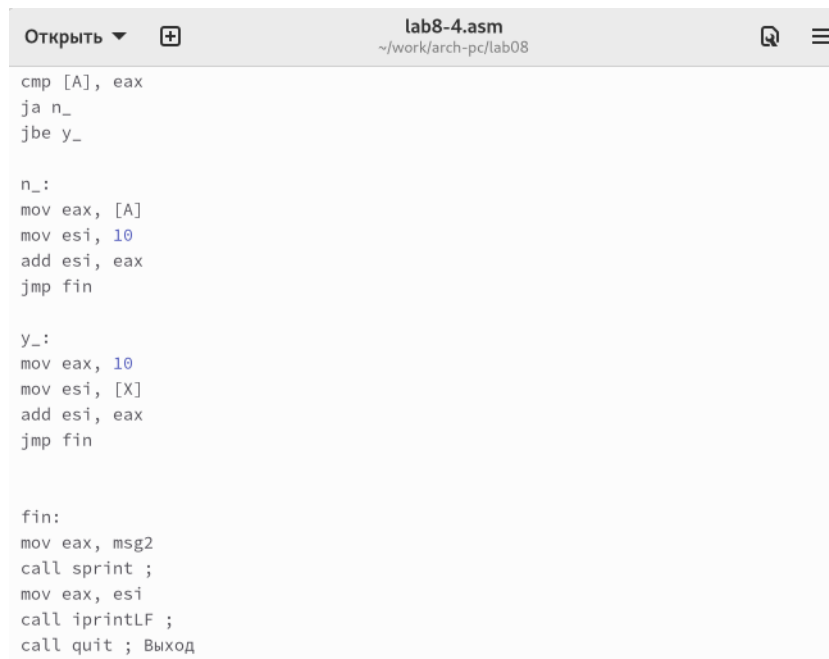
Рис. 3.17: lab8-3.asm




```
[fbnati@fedora lab08]$ nasm -f elf lab8-3.asm
[fbnati@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[fbnati@fedora lab08]$ ./lab8-3
Наименьшее число: 6
```

Рис. 3.18: Исполняемый файл

2. Написали программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. (рис. 3.19) Вид функции  $f(x)$  выбрали из таблицы вариантов заданий в соответствии с вариантом 15, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу для значений  $x$  и  $a$ . (рис. 3.20)



```
Открыть ▾  lab8-4.asm
~/.work/arch-pc/lab08

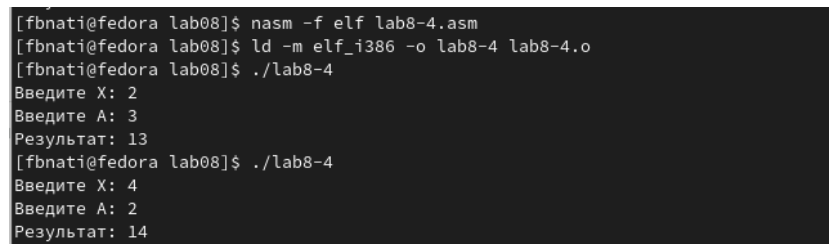
cmp [A], eax
ja n_
jbe y_

n_:
mov eax, [A]
mov esi, 10
add esi, eax
jmp fin

y_:
mov eax, 10
mov esi, [X]
add esi, eax
jmp fin

fin:
mov eax, msg2
call sprint ;
mov eax, esi
call iprintLF ;
call quit ; Выход
```

Рис. 3.19: lab8-4.asm



```
[fbnati@fedora lab08]$ nasm -f elf lab8-4.asm
[fbnati@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[fbnati@fedora lab08]$ ./lab8-4
Введите X: 2
Введите A: 3
Результат: 13
[fbnati@fedora lab08]$ ./lab8-4
Введите X: 4
Введите A: 2
Результат: 14
```

Рис. 3.20: Исполняемый файл

## 4 Выводы

В ходе выполнения лабораторной работы были изучены команды условного и безусловного переходов. Были приобретены навыки написания программ с использованием переходов. Ознакомились с назначением и структурой файла листинга.