

# **Очёт по лабораторной работе № 8**

**Архитектура Компьютера**

Старцева Алина Сергеевна

# 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 . . . . .	9
3.9	Текст программы . . . . .	10
3.10	Исполняемый файл . . . . .	10
3.11	Ключ -l . . . . .	11
3.12	mcedit . . . . .	11
3.13	lab8-2.lst . . . . .	11
3.14	lab8-2.asm . . . . .	12
3.15	mcedit . . . . .	12
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

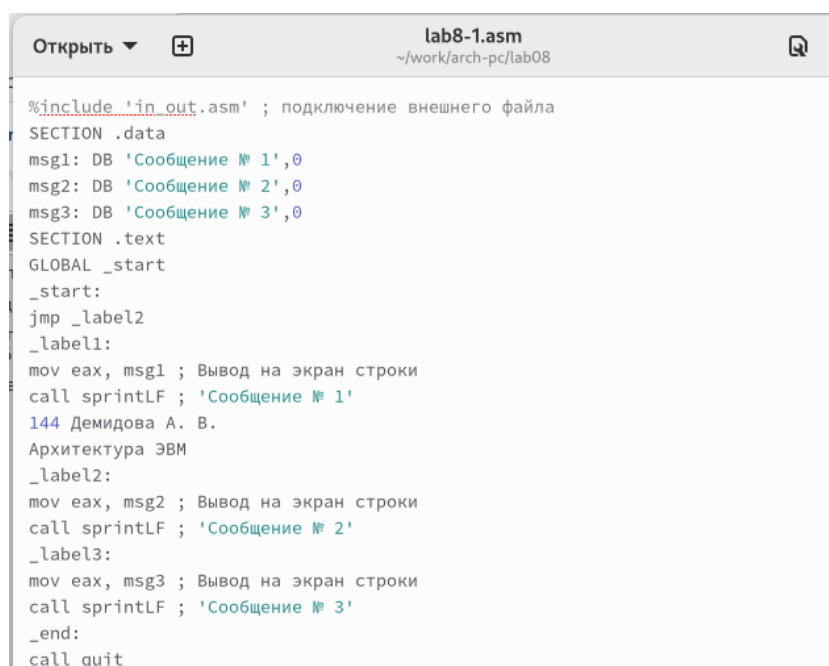
### 3.1 Реализация переходов в NASM

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

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

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)



```

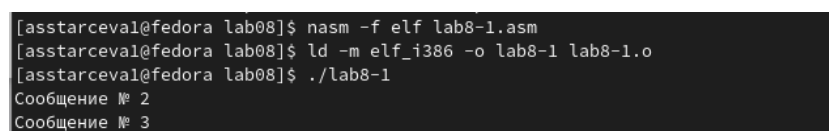
Открыть ▾ + lab8-1.asm ~/work/arch-pc/lab08

%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 sprintLF ; 'Сообщение № 1'
144 Демидова А. В.
Архитектура ЭВМ
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintLF ; 'Сообщение № 3'
_end:
call quit

```

Рис. 3.2: Текст программы

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



```

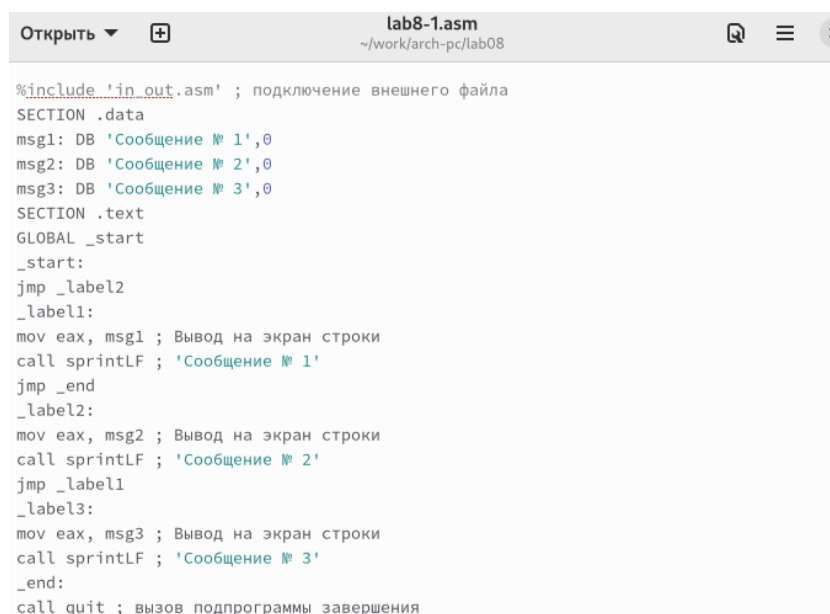
[asstarceval@fedora lab08]$ nasm -f elf lab8-1.asm
[asstarceval@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[asstarceval@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3

```

Рис. 3.3: Результат работы

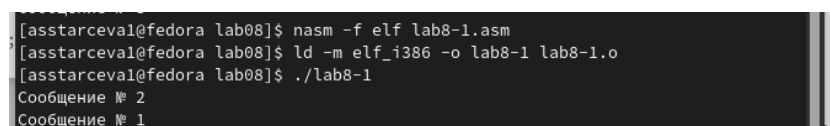
Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменили программу таким образом, чтобы 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)



```
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 printf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call printf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call printf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.4: Использование инструкций

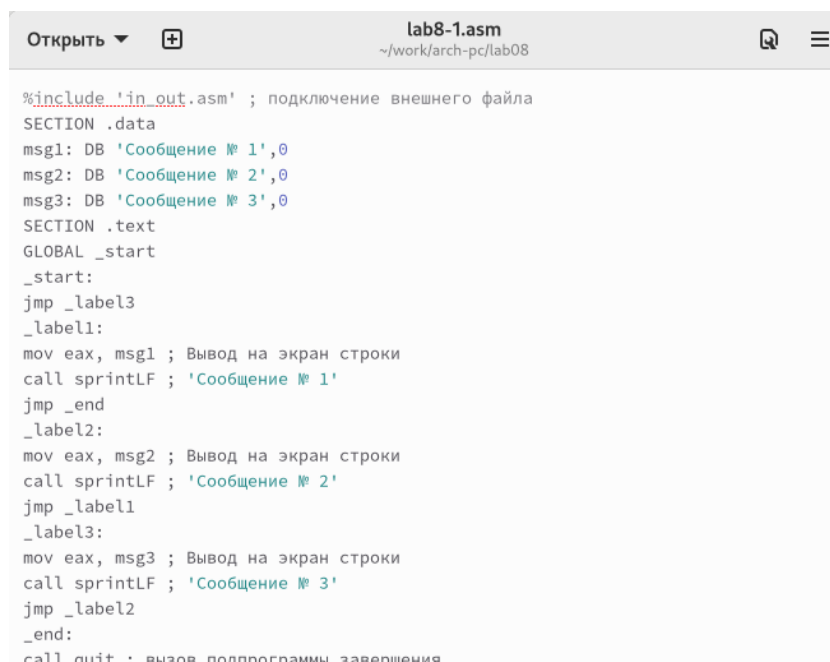


```
[asstarceval@fedora lab08]$ nasm -f elf lab8-1.asm
[asstarceval@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[asstarceval@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
```

Рис. 3.5: Текст программы

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

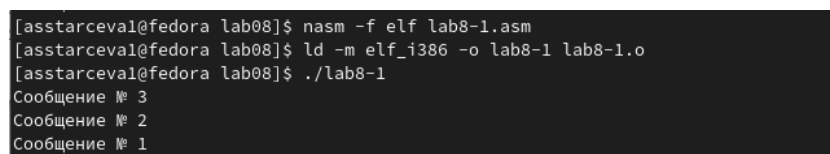




```
Открыть ▾ + lab8-1.asm
~/work/arch-pc/lab08

%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
```

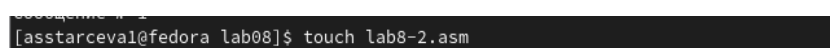
Рис. 3.6: Инструкции jmp



```
[asstarceval@fedora lab08]$ nasm -f elf lab8-1.asm
[asstarceval@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[asstarceval@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)



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

Рис. 3.8: lab8-2.asm

```

Открыть ▾  + lab8-2.asm
~/work/arch-pc/lab08

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'

```

Рис. 3.9: Текст программы

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

```

[asstarceval@fedora lab08]$ nasm -f elf lab8-2.asm
[asstarceval@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[asstarceval@fedora lab08]$ ./lab8-2
Введите B: 8
Наибольшее число: 50
[asstarceval@fedora lab08]$ ./lab8-2
Введите B: 69
Наибольшее число: 69

```

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)

```
[asstarceval@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)

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

Рис. 3.12: `mcedit`

```
asstarceval@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 00000000 <res Ah> max resb 10
8 0000000A <res Ah> B resb 10
9 00000000 section .text
10 00000000 global _start
11 00000000 _start:
12 00000000 ; ----- Вывод сообщения 'Введите B:
13 000000E8 B8[00000000] mov eax,msg1
14 000000ED E81DFFFFFF call sprint
15 000000F2 B9[0A000000] ; ----- Ввод 'B'
16 000000F7 BA0A000000 mov ecx,B
17 000000FC E842FFFFFF mov edx,10
18 00000101 B8[0A000000] call sread
19 00000106 E891FFFFFF ; ----- Преобразование 'B' из символа
20 0000010B A3[0A000000] mov eax,B
21 00000110 8B0D[35000000] call atoi ; Вызов подпрограммы перевода
22 00000114 B8[13000000] mov [B],eax ; запись преобразованного числа
23 00000118 8B0D[35000000] ; ----- Записываем 'A' в переменную
24 0000011C B8[35000000] mov ecx,[A] ; 'ecx = A'
25 00000120 8B0D[35000000]
```

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) Выполните трансляцию с получением файла листинга: (рис. 3.15), (рис. 3.16)

```
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,|
call sread
; ----- Преобразование 'В'
mov eax,B
call atoi : Вызов подпрограммы
```

Рис. 3.14: lab8-2.asm

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

Рис. 3.15: mcedit

```

asstarceva1@fedora:~/work/arch-pc/lab08 — mcedit lab8-2.lst
lab8-2.lst  [----]  0 L:[183+ 0 183/226] *(11182/14547b) 0032 0x020[*][X]
 8 00000000 <res Ah>          max resb 10
 9 0000000A <res Ah>          B resb 10
10                                section .text
11                                global _start
12                                _start:
13                                ; ----- Вывод сообщения 'Введите B:
14 000000E8 B8[00000000]      mov eax,msg1
15 000000ED E81DFFFFFF      call sprint
16                                ; ----- Ввод 'B'
17 000000F2 B9[0A000000]      mov ecx,B
18                                mov edx,
18                                *****
19 000000F7 E847FFFFFF      call sread
20                                ; ----- Преобразование 'B' из симво
21 000000FC B8[0A000000]      mov eax,B
22 00000101 E896FFFFFF      call atoi ; Вызов подпрограммы перевода
23 00000106 A3[0A000000]      mov [B],eax ; запись преобразованного чи
24                                ; ----- Записываем 'A' в переменную
25 0000010B 8B0D[35000000]    mov ecx,[A] ; 'ecx = A'
26 00000111 890D[00000000]    mov [max],ecx ; 'max = A'
27                                ; ----- Сравниваем 'A' и 'C' (как с
28 00000117 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'

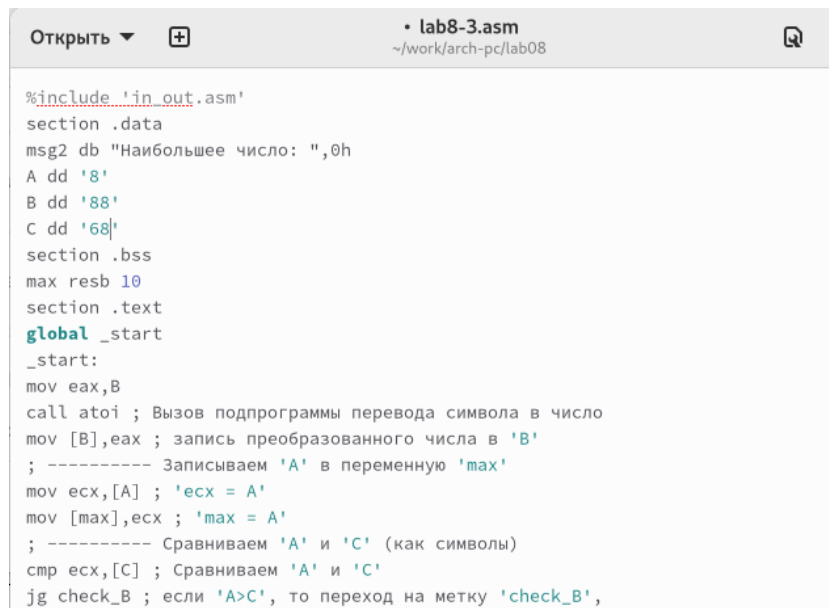
```

Рис. 3.16: lab8-2.lst

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

### 3.3 Задание для самостоятельной работы

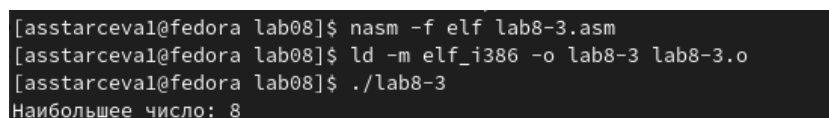
1. Написали программу нахождения наименьшей из 3 целочисленных переменных a, b и c. (рис. 3.17) Значения переменных выбрали из таблицы в соответствии с 4 вариантом, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу. (рис. 3.18)



```
Открыть ▾ + • lab8-3.asm
~/work/arch-pc/lab08

%include 'in_out.asm'
section .data
msg2 db "Наибольшее число: ",0h
A dd '8'
B dd '88'
C dd '68'
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'
jg check_B ; если 'A>C', то переход на метку 'check_B',
```

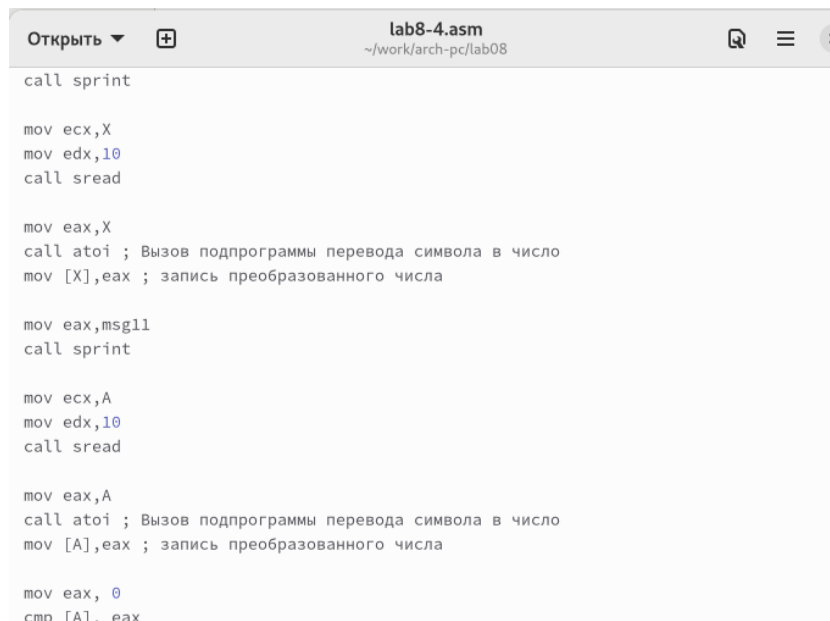
Рис. 3.17: lab8-3.asm



```
[asstarceval@fedora lab08]$ nasm -f elf lab8-3.asm
[asstarceval@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[asstarceval@fedora lab08]$ ./lab8-3
Наибольшее число: 8
```

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 4, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу для значений  $x$  и  $a$ . (рис. 3.20)



```
call sprint

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi ; Вызов подпрограммы перевода символа в число
mov [X],eax ; запись преобразованного числа

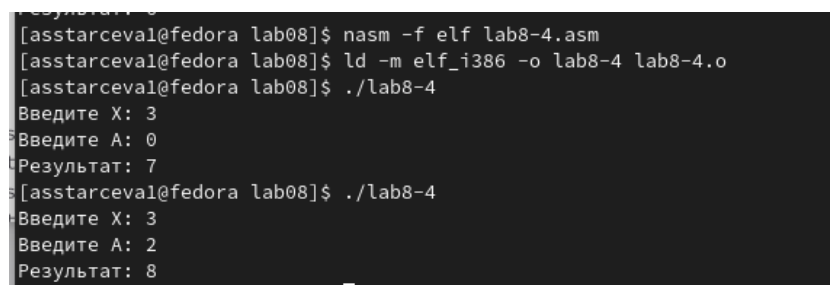
mov eax,msg11
call sprint

mov ecx,A
mov edx,10
call sread

mov eax,A
call atoi ; Вызов подпрограммы перевода символа в число
mov [A],eax ; запись преобразованного числа

mov eax, 0
cmp [A], eax
```

Рис. 3.19: lab8-4.asm



```
[asstarceval@fedora lab08]$ nasm -f elf lab8-4.asm
[asstarceval@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[asstarceval@fedora lab08]$ ./lab8-4
Введите X: 3
Введите A: 0
Результат: 7
[asstarceval@fedora lab08]$ ./lab8-4
Введите X: 3
Введите A: 2
Результат: 8
```

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.