

# **Отчет по лабораторной работе №7**

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

Мальянц Виктория Кареновна

# Содержание

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Цель работы</b>                                      | <b>6</b>  |
| <b>2</b> | <b>Задание</b>  | <b>7</b>  |
| <b>3</b> | <b>Выполнение лабораторной работы</b>                   | <b>8</b>  |
| 3.1      | Реализация переходов в NASM . . . . .                   | 8         |
| 3.2      | Изучение структуры файлы листинга . . . . .             | 13        |
| 3.3      | Выполнение заданий для самостоятельной работы . . . . . | 16        |
| <b>4</b> | <b>Выводы</b>   | <b>23</b> |

# Список иллюстраций

|      |  |    |
|------|--|----|
| 3.1  | Создание каталога и файла для программы . . . . .                | 8  |
| 3.2  | Копирование файла и просматривание содержимого каталога . . .    | 8  |
| 3.3  | Открытие файла lab7-1.asm в текстовом редакторе gedit . . . . .  | 8  |
| 3.4  | Редактирование файла . . . . .                                   | 9  |
| 3.5  | Запуск исполняемого файла . . . . .                              | 9  |
| 3.6  | Открытие файла lab7-1.asm в текстовом редакторе gedit . . . . .  | 9  |
| 3.7  | Редактирование файла . . . . .                                   | 10 |
| 3.8  | Запуск исполняемого файла . . . . .                              | 10 |
| 3.9  | Открытие файла lab7-1.asm в текстовом редакторе gedit . . . . .  | 10 |
| 3.10 | Редактирование файла . . . . .                                   | 11 |
| 3.11 | Запуск исполняемого файла . . . . .                              | 11 |
| 3.12 | Создание файла . . . . .   | 11 |
| 3.13 | Открытие файла lab7-2.asm в текстовом редакторе gedit . . . . .  | 11 |
| 3.14 | Редактирование файла . . . . .                                   | 12 |
| 3.15 | Редактирование файла . . . . .                                   | 12 |
| 3.16 | Запуск исполняемого файла . . . . .                              | 13 |
| 3.17 | Запуск исполняемого файла . . . . .                              | 13 |
| 3.18 | Запуск исполняемого файла . . . . .                              | 13 |
| 3.19 | Создание файла листинга . . . . .                                | 13 |
| 3.20 | Открытие файла lab7-2.lst в текстовом редакторе mcedit . . . . . | 13 |
| 3.21 | Строка под номером 15 . . . . .                                  | 14 |
| 3.22 | Строка под номером 19 . . . . .                                  | 14 |
| 3.23 | Строка под номером 22 . . . . .                                  | 14 |
| 3.24 | Открытие файла lab7-2.asm в текстовом редакторе gedit . . . . .  | 14 |
| 3.25 | Редактирование файла . . . . .                                   | 15 |
| 3.26 | Запуск исполняемого файла . . . . .                              | 15 |
| 3.27 | Открытие файла lab7-2.lst в текстовом редакторе mcedit . . . . . | 15 |
| 3.28 | Просмотр файла . . . . .   | 16 |
| 3.29 | Создание файла . . . . .   | 16 |
| 3.30 | Открытие файла lab7-3.asm в текстовом редакторе gedit . . . . .  | 16 |
| 3.31 | Редактирование файла . . . . .                                   | 17 |
| 3.32 | Запуск исполняемого файла . . . . .                              | 17 |
| 3.33 | Создание файла . . . . .   | 19 |
| 3.34 | Открытие файла lab7-4.asm в текстовом редакторе gedit . . . . .  | 19 |
| 3.35 | Редактирование файла . . . . .                                   | 20 |
| 3.36 | Запуск исполняемого файла . . . . .                              | 20 |

|  |    |
|--|----|
| 3.37 Запуск исполняемого файла . . . . . | 20 |
|--|----|

## **Список таблиц**

# 1 Цель работы

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

## **2 Задание**

1. Реализация переходов в NASM
2. Изучение структуры файлы листинга
3. Выполнение заданий для самостоятельной работы

## 3 Выполнение лабораторной работы

### 3.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 3.1).

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

Рис. 3.1: Создание каталога и файла для программы

С помощью команды `cp` копирую файл `in_out.asm` и просматриваю содержимое каталога `lab07` с помощью команды `ls` (рис. 3.2).

```
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ cp ~/3арпузки/in_out.asm in_out.asm
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm
```

Рис. 3.2: Копирование файла и просмотр содержимого каталога

Открываю файл `lab7-1.asm` в текстовом редакторе `gedit` через терминал (рис. 3.3).

```
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ gedit lab7-1.asm
```

Рис. 3.3: Открытие файла `lab7-1.asm` в текстовом редакторе `gedit`

Ввожу в файл `lab7-1.asm` программу с использованием инструкции `jmp` (рис. 3.4).



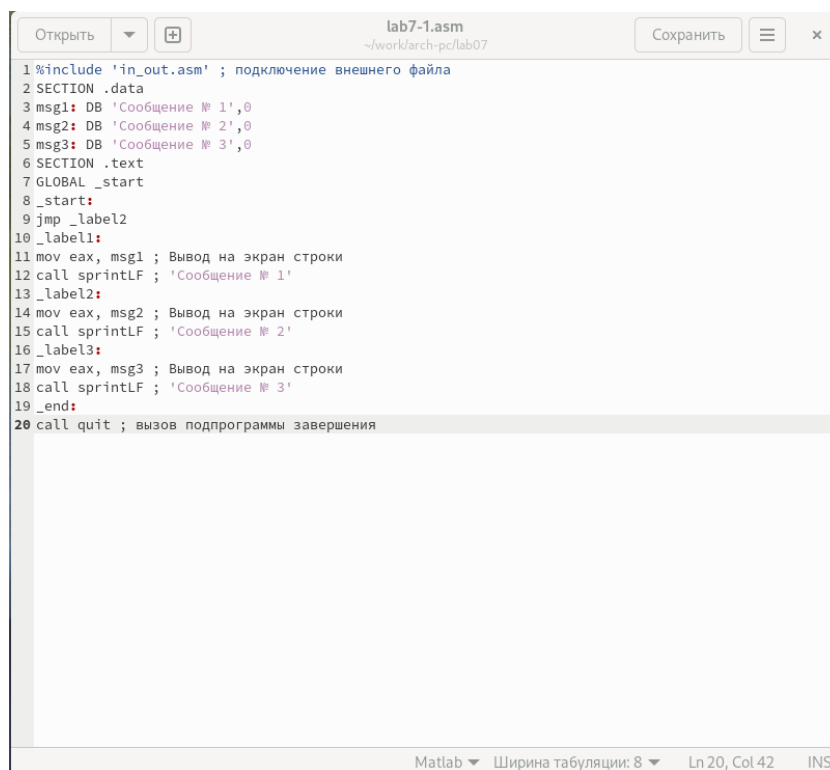


Рис. 3.4: Редактирование файла

Создаю исполняемый файл и запускаю его. Убеждаюсь в том, что безусловный переход изменяет порядок выполнения инструкций (рис. 3.5).

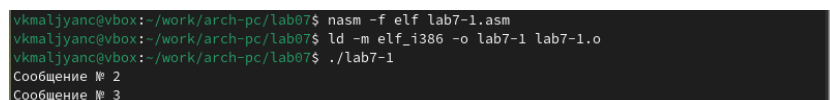


Рис. 3.5: Запуск исполняемого файла

Открываю файл lab7-1.asm в текстовом редакторе gedit через терминал (рис. 3.6).

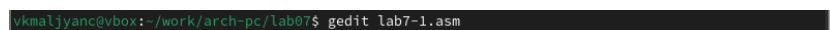


Рис. 3.6: Открытие файла lab7-1.asm в текстовом редакторе gedit

Изменяю программу таким образом, чтобы сначала выводилось “Сообщение № 2”, а затем “Сообщение № 3” (рис. 3.7).

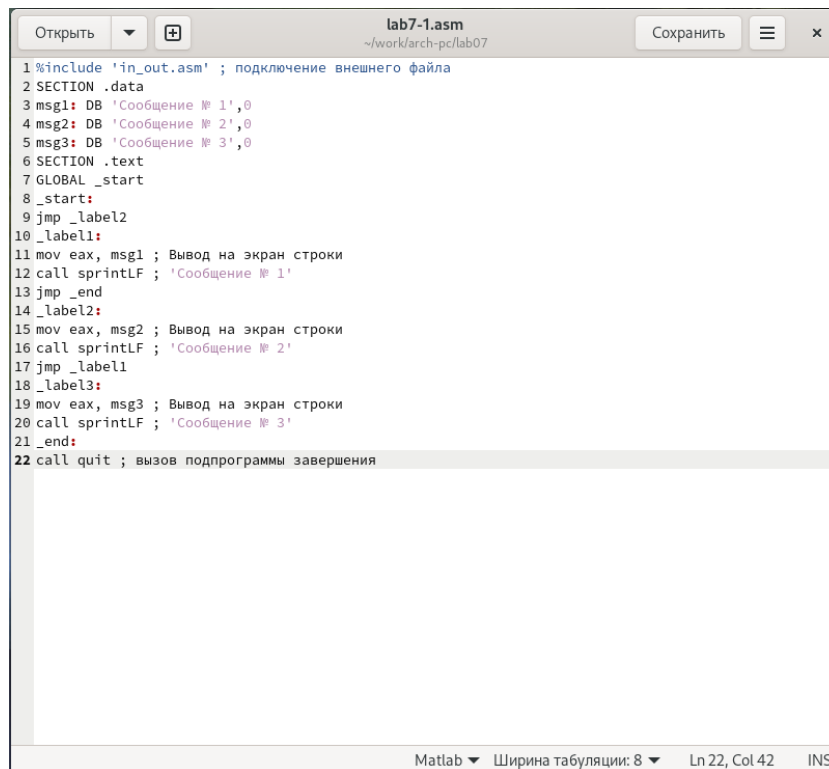


Рис. 3.7: Редактирование файла

Создаю исполняемый файл и запускаю его. Убеждаюсь в том, что изменения применены корректно (рис. 3.8).

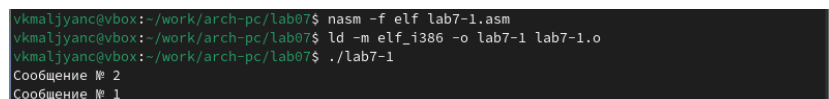


Рис. 3.8: Запуск исполняемого файла

Открываю файл lab7-1.asm в текстовом редакторе gedit через терминал (рис. 3.9).

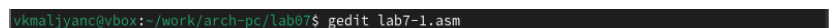


Рис. 3.9: Открытие файла lab7-1.asm в текстовом редакторе gedit

Изменяю программу таким образом, чтобы сначала выводилось “Сообщение № 3”, потом “Сообщение № 2”, а затем “Сообщение № 1” (рис. 3.10).

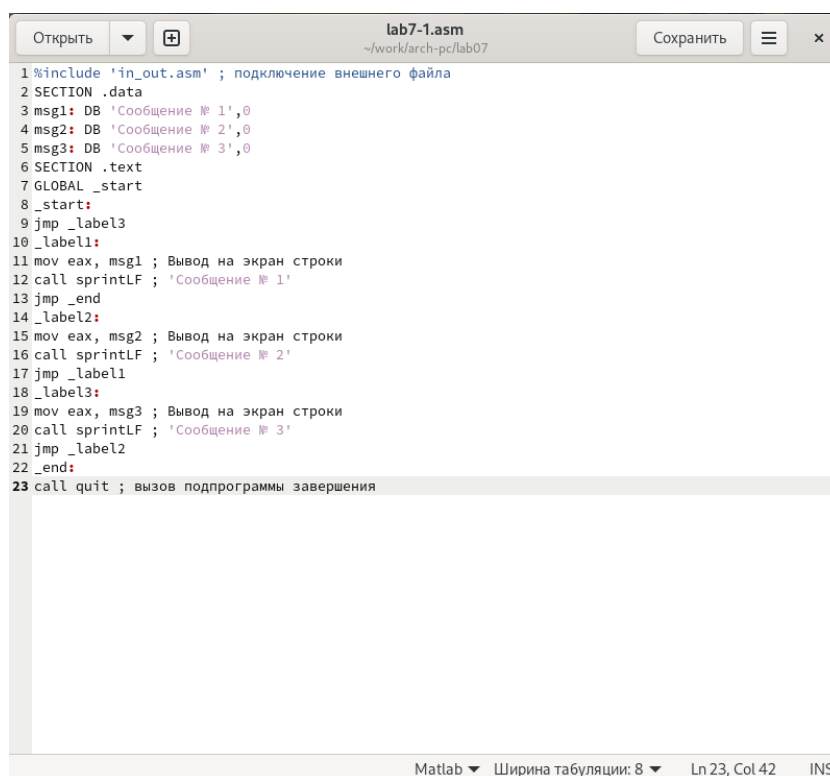


Рис. 3.10: Редактирование файла

Создаю исполняемый файл и запускаю его. Убеждаюсь в том, что изменения применены корректно (рис. 3.11).

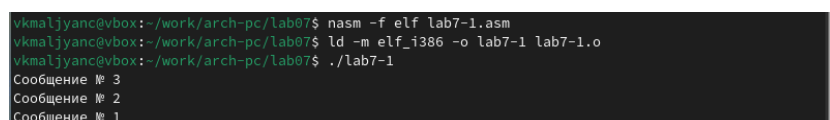


Рис. 3.11: Запуск исполняемого файла

С помощью команды touch создаю файл lab7-2.asm (рис. 3.12).

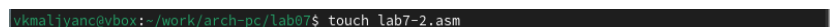


Рис. 3.12: Создание файла

Открываю файл lab7-2.asm в текстовом редакторе gedit через терминал (рис. 3.13).

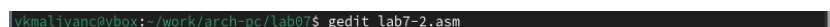


Рис. 3.13: Открытие файла lab7-2.asm в текстовом редакторе gedit

Ввожу в файл lab7-2.asm программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C (рис. 3.14) (рис. 3.15).

```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)

```

Рис. 3.14: Редактирование файла

```

38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprint ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call iprintLF ; Вывод 'max(A,B,C)'
49 call quit ; Выход

```

Рис. 3.15: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла для значения B равного 10 (рис. 3.16), B равного 30 (рис. 3.17) и B равного 60 (рис. 3.18). Убеждаюсь в том, что программа корректно выводит на экран наибольшую из 3 целочисленных переменных: A, B и C.

```

vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50

```

Рис. 3.16: Запуск исполняемого файла

```

vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50

```

Рис. 3.17: Запуск исполняемого файла

```

vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ ./lab7-2
Введите B: 60
Наибольшее число: 60

```

Рис. 3.18: Запуск исполняемого файла

## 3.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.lst, указав ключ -l и задав имя файла листинга в командной строке (рис. 3.19)

```

vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm

```

Рис. 3.19: Создание файла листинга

Открываю файл lab7-2.lst в текстовом редакторе mcedit через терминал (рис. 3.20).

```

vkmal'jyanc@vbox: ~/work/arch-pc/lab07$ mcedit lab7-2.lst

```

Рис. 3.20: Открытие файла lab7-2.lst в текстовом редакторе mcedit

Объяснение содержимого трех строк файла листинга:

1. Строка под номером 15 (рис. 3.21). Первое значение - номер строки (15), второе вхождение - адрес (000000ED), третье вхождение - машинный код (E81DFFFFFF, из которого E8 - префикс команды call, 1DFFFFFF - смещение до адреса подпрограммы sprint), четвертое вхождение - инструкция (call sprint - команда, которая вызывает функцию sprint).

```
15 000000ED E81DFFFFFF call sprint
```

Рис. 3.21: Строка под номером 15

2. Строка под номером 19 (рис. 3.22). Первое значение - номер строки (19), второе вхождение - адрес (000000FC), третье вхождение - машинный код (E842FFFFFF, из которого E8 - префикс команды call, 42FFFFFF - смещение до адреса подпрограммы sread), четвертое вхождение - инструкция (call sread - команда, которая вызывает функцию sread).

```
19 000000FC E842FFFFFF call sread
```

Рис. 3.22: Строка под номером 19

3. Строка под номером 22 (рис. 3.23). Первое значение - номер строки (22), второе вхождение - адрес (00000106), третье вхождение - машинный код (E891FFFFFF, из которого E8 - префикс команды call, 91FFFFFF - смещение до адреса подпрограммы atoi), четвертое вхождение - инструкция (call atoi - команда, которая вызывает функцию atoi).

```
22 00000106 E891FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
```

Рис. 3.23: Строка под номером 22

Открываю файл lab7-2.asm в текстовом редакторе gedit через терминал (рис. 3.24).

```
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ gedit lab7-2.asm
```

Рис. 3.24: Открытие файла lab7-2.asm в текстовом редакторе gedit

Удаляю из строки 38 операнд [max] (рис. 3.25).

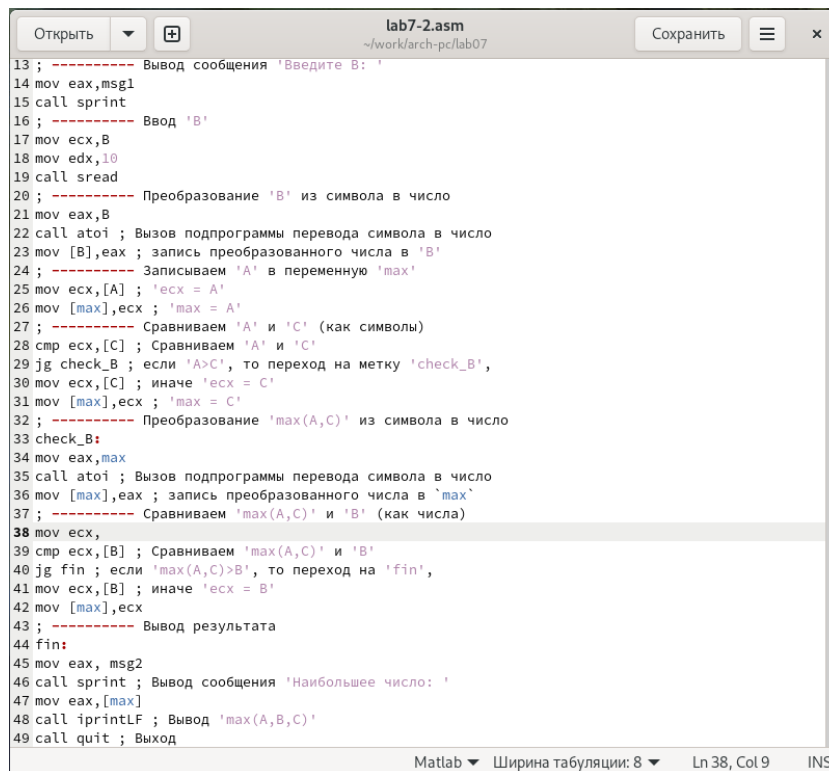


Рис. 3.25: Редактирование файла

Выполняю трансляцию с получением файла листинга (рис. 3.26). Никакие выходные файла кроме листинга не создаются.

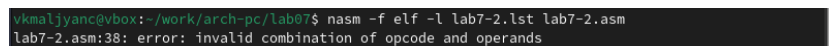


Рис. 3.26: Запуск исполняемого файла

Открываю файл lab7-2.lst в текстовом редакторе mcedit через терминал (рис. 3.27).

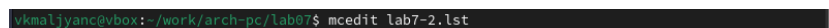


Рис. 3.27: Открытие файла lab7-2.lst в текстовом редакторе mcedit

В листинге добавляется ошибка (рис. 3.28).

```

lab7-2.lst      [----] 48 L:[194+19 213/226] *(13625/14544b) 0010 0x00A      [*][X]
19 000000FC E842FFFFFF      call sread
20                               ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000]      mov [B],eax ; запись преобразованного числа в 'B'
24                               ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000]      mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000]      mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000]      cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C              jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000]      mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000]      mov [max],ecx ; 'max = C'
32                               ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000]      mov eax,max
35 00000135 E862FFFFFF      call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000]      mov [max],eax ; запись преобразованного числа в 'max'
37                               ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 mov ecx,[max]
39                               .error: invalid combination of opcode and operands
39 0000013F 3B0D[0A000000]      cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000145 7F0C              jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000147 8B0D[0A000000]      mov ecx,[B] ; иначе 'ecx = B'
42 0000014D 890D[00000000]      mov [max],ecx
43                               ; ----- Вывод результата
44 fin:
45 00000153 B8[13000000]      mov eax, msg2
46 00000158 E882FFFFFF      call sprintf ; Вывод сообщения 'Наибольшее число: '
47 0000015D A1[00000000]      mov eax,[max]
48 00000162 E81FFFFFFF      call iprintLF ; Вывод 'max(A,B,C)'
49 00000167 E86FFFFFFF      call quit ; Выход

```

Рис. 3.28: Просмотр файла

## 3.3 Выполнение заданий для самостоятельной работы

Задание № 1. С помощью команды touch создаю файл lab7-3.asm (рис. 3.29).

```

vkmaliyanc@vbox:~/work/arch-pc/lab07$ touch lab7-3.asm

```

Рис. 3.29: Создание файла

Открываю файл lab7-3.asm в текстовом редакторе gedit через терминал (рис. 3.30).

```

vkmaliyanc@vbox:~/work/arch-pc/lab07$ gedit lab7-3.asm

```

Рис. 3.30: Открытие файла lab7-3.asm в текстовом редакторе gedit

Ввожу в файл lab7-3.asm программу для нахождения наименьшей из 3 целочисленных переменных a,b и c. Ввожу функцию из варианта №1 (рис. 3.31).



```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наименьшее число: ",0h
5 A dd '17'
6 C dd '45'
7 section .bss
8 min resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'min'
25 mov ecx,[A] ; 'ecx = A'
26 mov [min],ecx ; 'min = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A<C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [min],ecx ; 'min = C'
32 ; ----- Преобразование 'min(A,C)' из символа в число
33 check_B:
34 mov eax,min
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [min],eax ; запись преобразованного числа в 'min'
37 ; ----- Сравниваем 'min(A,C)' и 'B' (как числа)

```

Рис. 3.31: Редактирование файла

Создаю исполняемый файл и запускаю его. Задаю значение B равное 23 (рис. 3.32). Убеждаюсь в том, что результат выводится корректно.

```

vkmalijanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
vkmalijanc@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
vkmalijanc@vbox: ~/work/arch-pc/lab07$ ./lab7-3
Введите B: 23
Наименьшее число: 17

```

Рис. 3.32: Запуск исполняемого файла

Листинг программы:

```

#include 'in_out.asm'

section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '17'
C dd '45'

```

```

section .bss
min 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 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = 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 [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число

```

```

mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jb fin ; если 'min(A,C)<B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наименьшее число: '
mov eax,[min]
call iprintLF ; Вывод 'min(A,B,C)'
call quit ; Выход

```

Задание № 2. С помощью команды touch создаю файл lab7-4.asm (рис. 3.33).



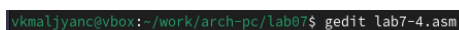
```

vkmalijanc@vbox: ~/work/arch-pc/lab07$ touch lab7-4.asm

```

Рис. 3.33: Создание файла

Открываю файл lab7-4.asm в текстовом редакторе gedit через терминал (рис. 3.34).



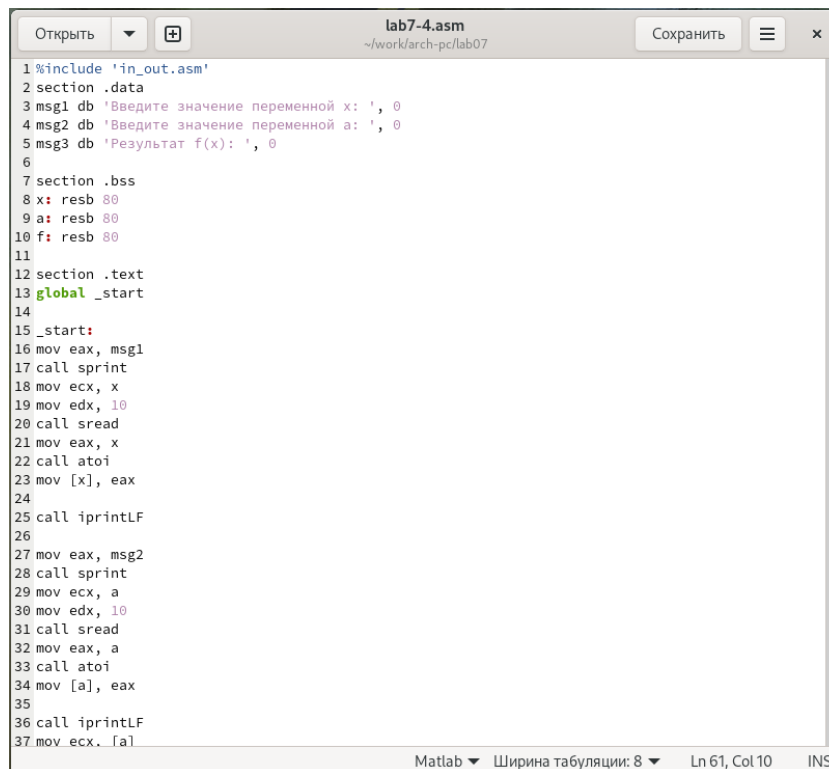
```

vkmalijanc@vbox: ~/work/arch-pc/lab07$ gedit lab7-4.asm

```

Рис. 3.34: Открытие файла lab7-4.asm в текстовом редакторе gedit

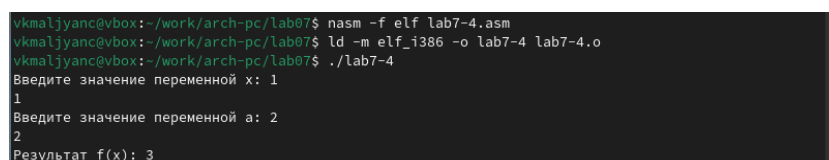
Ввожу в файл lab7-4.asm программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений. Ввожу функцию из варианта №1 (рис. 3.35).



```
1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите значение переменной x: ', 0
4 msg2 db 'Введите значение переменной a: ', 0
5 msg3 db 'Результат f(x): ', 0
6
7 section .bss
8 x: resb 80
9 a: resb 80
10 f: resb 80
11
12 section .text
13 global _start
14
15 _start:
16 mov eax, msg1
17 call sprint
18 mov ecx, x
19 mov edx, 10
20 call sread
21 mov eax, x
22 call atoi
23 mov [x], eax
24
25 call iprintLF
26
27 mov eax, msg2
28 call sprint
29 mov ecx, a
30 mov edx, 10
31 call sread
32 mov eax, a
33 call atoi
34 mov [a], eax
35
36 call iprintLF
37 mov ecx, [a]
```

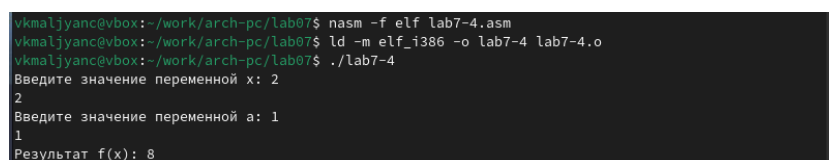
Рис. 3.35: Редактирование файла

Создаю исполняемый файл и запускаю его. Проверяю работу исполняемого файла для значения x равного 1 и a равного 2 (рис. 3.36) и x равного 2 и a равного 1 (рис. 3.37). Убеждаюсь в том, что результат программы выводится корректно.



```
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 1
1
Введите значение переменной a: 2
2
Результат f(x): 3
```

Рис. 3.36: Запуск исполняемого файла



```
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
vkmaljyanc@vbox: ~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 2
2
Введите значение переменной a: 1
1
Результат f(x): 8
```

Рис. 3.37: Запуск исполняемого файла

Листинг программы:

```

%include 'in_out.asm'

section .data
msg1 db 'Введите значение переменной x: ', 0
msg2 db 'Введите значение переменной a: ', 0
msg3 db 'Результат f(x): ', 0

section .bss
x: resb 80
a: resb 80
f: resb 80

section .text
global _start

_start:
mov eax, msg1
call sprint
mov ecx, x
mov edx, 10
call sread
mov eax, x
call atoi
mov [x], eax

call iprintLF

mov eax, msg2
call sprint
mov ecx, a

```

```

mov edx, 10
call sread
mov eax, a
call atoi
mov [a], eax

call iprintLF
mov ecx, [a]
cmp ecx, [x]
jge less

mov edx, 8
mov [f], edx
jmp fin

less:
mov ebx, [x]
mov ax, 2
mul ax
sub eax, ebx
mov [f], eax

fin:
mov eax, msg3
call sprint
mov eax, [f]
call iprintLF
call quit

```

## **4 Выводы**

Я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Познакомилась с назначением и структурой файла листинга.