

**Отчет по лабораторной работе №7**  
**Номер студенческого билета:**  
**10322536339**

Габдуллина Александра Булатовна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>4</b>	<b>Выводы</b>	<b>15</b>

## Список иллюстраций

3.1	Создание каталога, переход в каталог и создание файла lab7-1.asm . . .	7
3.2	Вводим текст программы в файл lab7-1.asm . . . . .	7
3.3	Создание файла и запуск . . . . .	8
3.4	Изменение текста программы . . . . .	8
3.5	Создание файла и проверка его работы . . . . .	8
3.6	Изменяем текст программы . . . . .	9
3.7	Создание файла и проверка его работы . . . . .	9
3.8	Создание файла lab7-2.asm . . . . .	9
3.9	Создание, запуск и проверка файла lab7-2.asm для значений 5 и 100 . .	10
3.10	Создание файла листинга для программы из файла и открывание его .	10
3.11	Файл lab7-2.lst . . . . .	11
3.12	Удаление операнда в 31 строке в файле . . . . .	12
3.13	Выполняем трансляцию . . . . .	12
3.14	Проверяем, какие файлы у нас создались . . . . .	12
3.15	Строка 31 в листинге . . . . .	13
3.16	Текст программы задания 1 из самостоятельной работы . . . . .	13
3.17	Создание файла и проверка его работы . . . . .	13
3.18	Текст программы задания 2 из самостоятельной работы . . . . .	14
3.19	Создание файла и проверка его работы . . . . .	14

## **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

В качестве ответа на лабораторную работу в ТУИС необходимо загрузить :

1.Ссылка на репозиторий (<https://github.com/>) 2.Следует загрузить отдельными файлами (не общим архивом): 3.отчёт в markdown; 4.отчёт в docx (сделанный из markdown); 5.отчёт в pdf (сделанный из markdown); 6.архив с исходными материалами markdown (текстовые файлы, скриншоты и т. д.) 7.Файл(ы) с исходным текстом программ, написанных в ходе выполнения работы (в формате .asm).

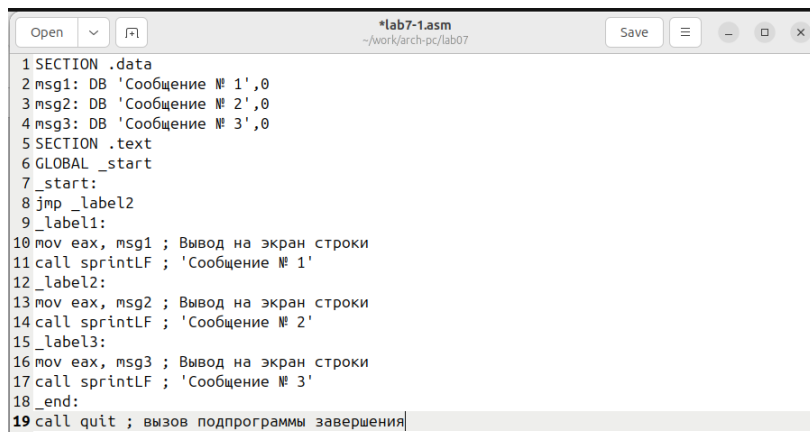
### 3 Выполнение лабораторной работы

Создаём каталог для программ лабораторной работы № 7, переходим в него и создаём файл lab7-1.asm:

```
gabdu1linaab@ubuntu:~$ mkdir ~/work/arch-pc/lab07
gabdu1linaab@ubuntu:~$ cd ~/work/arch-pc/lab07
gabdu1linaab@ubuntu:~/work/arch-pc/lab07$ touch lab7-1.asm
```

Рисунок 3.1: Создание каталога, переход в каталог и создание файла lab7-1.asm

Введём в файл lab7-1.asm текст программы из листинга 7.1.



```
*lab7-1.asm
~/work/arch-pc/lab07
Save

1 SECTION .data
2 msg1: DB 'Сообщение № 1',0
3 msg2: DB 'Сообщение № 2',0
4 msg3: DB 'Сообщение № 3',0
5 SECTION .text
6 GLOBAL _start
7 _start:
8 jmp _label2
9 _label1:
10 mov eax, msg1 ; Вывод на экран строки
11 call sprintf ; 'Сообщение № 1'
12 _label2:
13 mov eax, msg2 ; Вывод на экран строки
14 call sprintf ; 'Сообщение № 2'
15 _label3:
16 mov eax, msg3 ; Вывод на экран строки
17 call sprintf ; 'Сообщение № 3'
18 _end:
19 call quit ; вызов подпрограммы завершения
```

Рисунок 3.2: Вводим текст программы в файл lab7-1.asm

Создаём исполняемый файл и запускаем его.

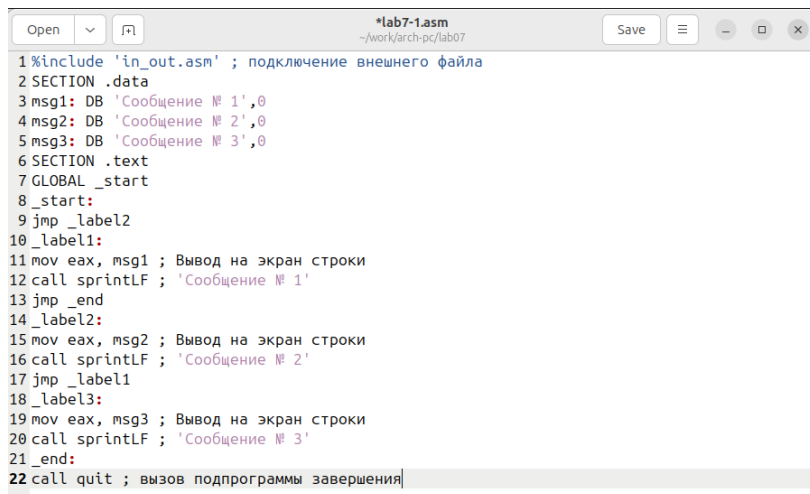
```

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

```

Рисунок 3.3: Создание файла и запуск

Изменяем текст программы в соответствии с листингом 7.2.



```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label2
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения

```

Рисунок 3.4: Изменение текста программы

Создаём исполняемый файл и проверяем его работу.

```

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

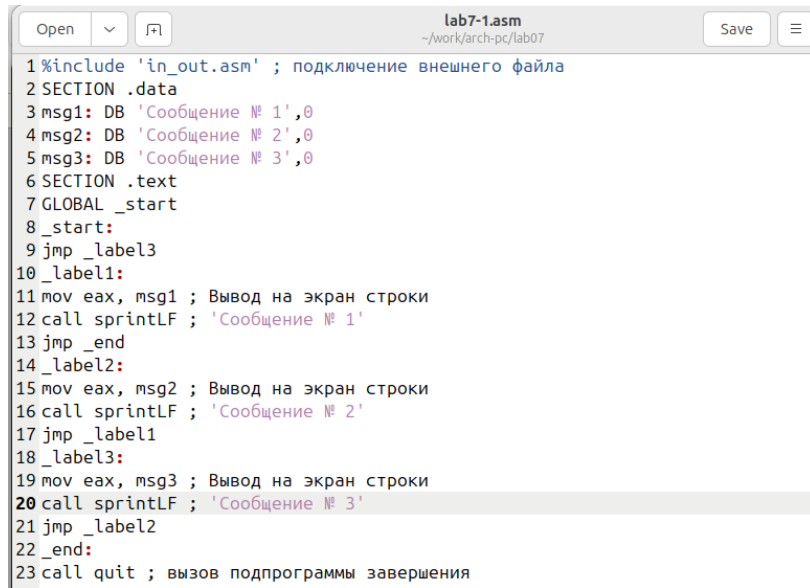
```

Рисунок 3.5: Создание файла и проверка его работы

Снова изменяем текст программы, чтобы вывод программы был следующим:

Сообщение № 3  
Сообщение № 2  
Сообщение № 1

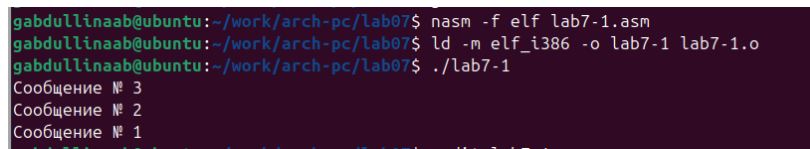




```
lab7-1.asm
~/work/arch-pc/lab07
1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg1: DB 'Сообщение № 1',0
4 msg2: DB 'Сообщение № 2',0
5 msg3: DB 'Сообщение № 3',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 jmp _label3
10 _label1:
11 mov eax, msg1 ; Вывод на экран строки
12 call sprintf ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 3'
21 jmp _label2
22 _end:
23 call quit ; вызов подпрограммы завершения
```

Рисунок 3.6: Изменяем текст программы

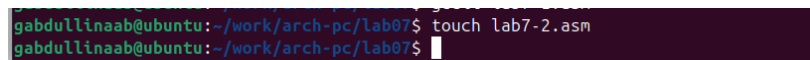
Создаём исполняемый файл и проверяем его работу.



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

Рисунок 3.7: Создание файла и проверка его работы

Создаём файл lab7-2.asm в каталоге ~/work/arch-pc/lab07.



```
gabduullinaab@ubuntu:~/work/arch-pc/lab07$ touch lab7-2.asm
gabduullinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.8: Создание файла lab7-2.asm

Вводим текст программы из листинга 7.3 в файл в lab7-2.asm, запускаем файл и проверяем его работу для различных значений В.

```

gabdullinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 5
Наибольшее число: 50
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 100
Наибольшее число: 100
gabdullinaab@ubuntu:~/work/arch-pc/lab07$

```

Рисунок 3.9: Создание, запуск и проверка файла lab7-2.asm для значений 5 и 100

Создаём файл листинга для программы из файла lab7-2.asm и с помощью текстового редактора mcedit открываем его.

```

gabdullinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ mcedit lab7-2.lst

```

Рисунок 3.10: Создание файла листинга для программы из файла и открывание его

Ознакомимся с его форматом и содержанием. Объяснение содержимого трёх строк файла листинга по выбору:

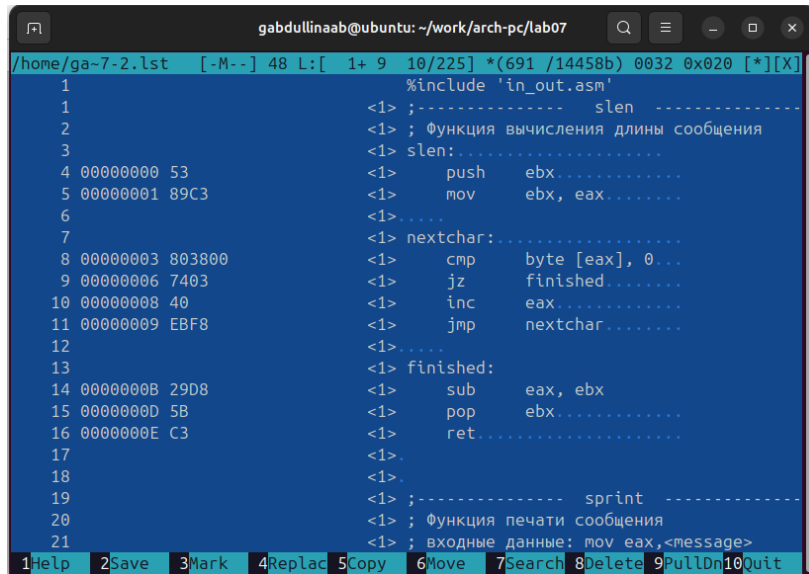
Строка 5 push ebx - инструкция сохраняет значение регистра EBX в стек Машинный код: 53 - шестнадцатеричное представление инструкции PUSH EBX Адрес: 00000000 - относительный адрес в памяти, где будет расположена эта инструкция <1> - номер исходного файла (основной файл)

Строка 8 cmp byte [eax], 0 - сравнивает байт по адресу в EAX с нулём Машинный код: 803800 - состоит из: 80 - код операции CMP 38 - модификатор для сравнения байта в памяти 00 - значение для сравнения (ноль) Адрес: 00000003 - позиция в памяти

Проверяем, не достигли ли мы конца строки (строки в С заканчиваются нулевым байтом)

Строка 9 jz finished - переход к метке finished, если предыдущее сравнение было равно 0 Машинный код: 7403 - 74 = код JZ, 03 = смещение на 3 байта вперёд Адрес: 00000006

Если найден нулевой байт(конец строки), выходим из цикла .



```
gabduallinaab@ubuntu: ~/work/arch-pc/lab07
/home/ga-7-2.lst [-M--] 48 L: [ 1+ 9 10/225] *(691 /14458b) 0032 0x020 [*][X]
1                                     %include 'in_out.asm'
1                                     <1> ;----- slen -----
2                                     <1> ; Функция вычисления длины сообщения
3                                     <1> slen:-----
4 00000000 53                         <1> push    ebx
5 00000001 89C3                       <1> mov     ebx, eax
6                                     <1> .....
7                                     <1> nextchar:-----
8 00000003 803800                     <1> cmp     byte [eax], 0
9 00000006 7403                       <1> jz      finished
10 00000008 40                        <1> inc     eax
11 00000009 EBF8                      <1> jmp     nextchar
12                                     <1> .....
13                                     <1> finished:
14 0000000B 29D8                     <1> sub     eax, ebx
15 0000000D 5B                        <1> pop     ebx
16 0000000E C3                       <1> ret
17                                     <1> .
18                                     <1> .
19                                     <1> ;----- sprint -----
20                                     <1> ; Функция печати сообщения
21                                     <1> ; входные данные: mov eax,<message>
```

Рисунок 3.11: Файл lab7-2.lst

В файле lab7-2.asm в строке 31 нужно удалить второй операнд:

Было:

31. mov [max],ecx ; „max = C“

Стало:

31. mov [max] ; „max = C“

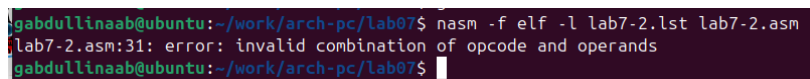
Открываем файл с программой lab7-2.asm и в 31 строке удаляем один операнд.



```
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)
38 cmp ecx,[max]
```

Рисунок 3.12: Удаление операнда в 31 строке в файле

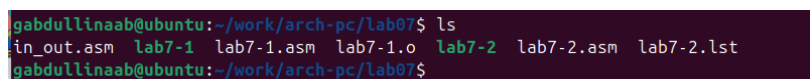
Выполняем трансляцию с получением файла листинга.



```
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:31: error: invalid combination of opcode and operands
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.13: Выполняем трансляцию

С помощью команды ls проверим какие файлы у нас создались.



```
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.14: Проверяем, какие файлы у нас создались

У нас создались:

- 1) lab7-2.lst - листинг (пустой или с ошибкой)
- 2) Объектный файл lab7-2.o НЕ создался - потому что трансляция завершилась с ошибкой

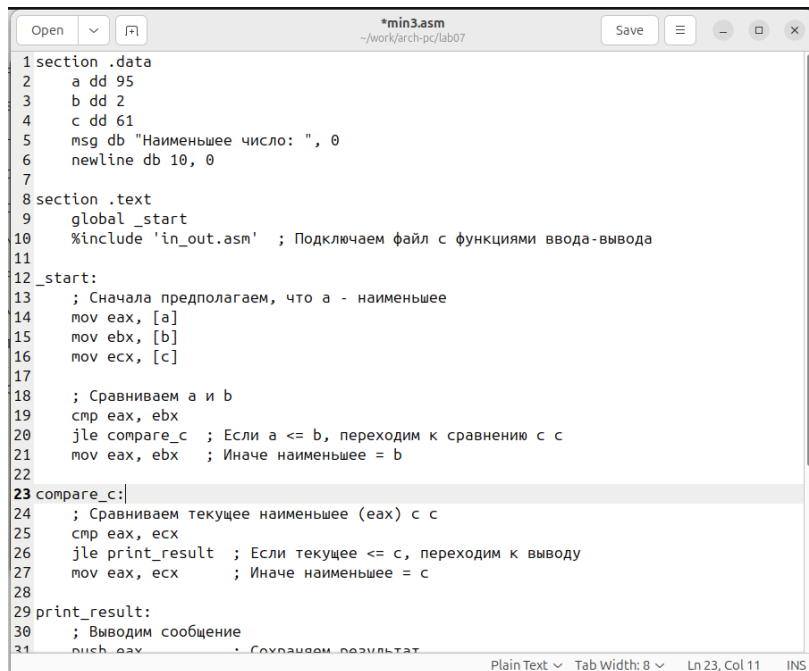
Посмотрим, что у нас добавилось в листинге.

```
31 ***** error: invalid combination of opcode and operands
```

Рисунок 3.15: Строка 31 в листинге

### #Задание для самостоятельной работы

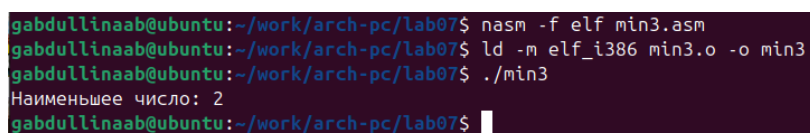
Напишем программу нахождения наименьшей из 3 целочисленных переменных 95,2,61 (вариант 20) .



```
1 section .data
2     a dd 95
3     b dd 2
4     c dd 61
5     msg db "Наименьшее число: ", 0
6     newline db 10, 0
7
8 section .text
9     global _start
10    %include 'in_out.asm' ; Подключаем файл с функциями ввода-вывода
11
12 _start:
13     ; Сначала предполагаем, что a - наименьшее
14     mov eax, [a]
15     mov ebx, [b]
16     mov ecx, [c]
17
18     ; Сравниваем a и b
19     cmp eax, ebx
20     jle compare_c ; Если a <= b, переходим к сравнению с c
21     mov eax, ebx ; Иначе наименьшее = b
22
23 compare_c:
24     ; Сравниваем текущее наименьшее (eax) с c
25     cmp eax, ecx
26     jle print_result ; Если текущее <= c, переходим к выводу
27     mov eax, ecx ; Иначе наименьшее = c
28
29 print_result:
30     ; Выводим сообщение
31     push eax ; Сохраняем результат
```

Рисунок 3.16: Текст программы задания 1 из самостоятельной работы

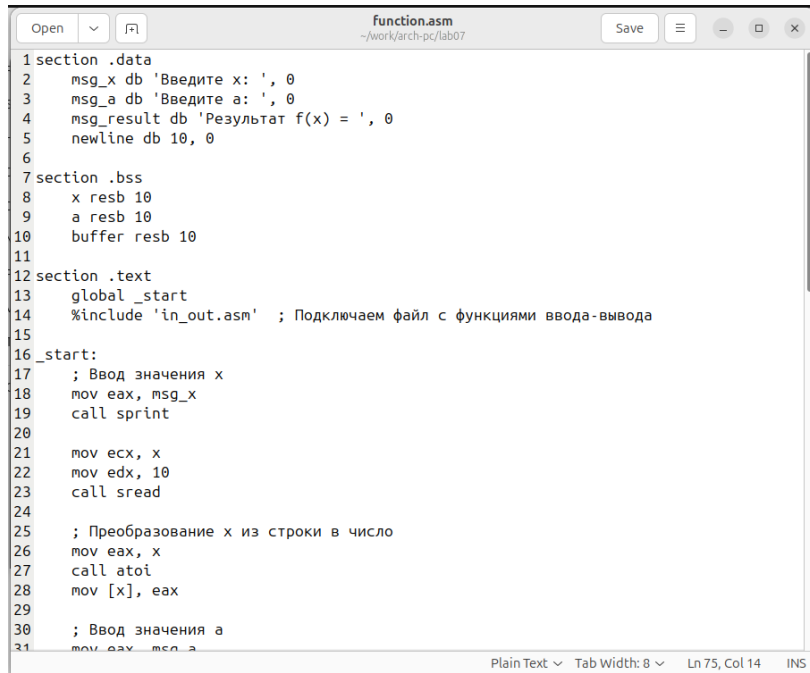
Создаём исполняемый файл и проверяем его работу.



```
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf min3.asm
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 min3.o -o min3
gabdullinaab@ubuntu:~/work/arch-pc/lab07$ ./min3
Наименьшее число: 2
gabdullinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.17: Создание файла и проверка его работы

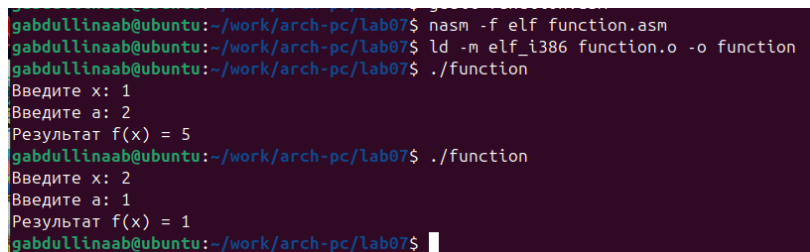
Напишем программу, которая для введенных с клавиатуры значений и вычисляет значение заданной функции ( ) и выводит результат вычислений. Вид функции ( ) выбрала из таблицы 7.6. (вариант 20)



```
1 section .data
2     msg_x db 'Введите x: ', 0
3     msg_a db 'Введите a: ', 0
4     msg_result db 'Результат f(x) = ', 0
5     newline db 10, 0
6
7 section .bss
8     x resb 10
9     a resb 10
10    buffer resb 10
11
12 section .text
13     global _start
14     %include 'in_out.asm' ; Подключаем файл с функциями ввода-вывода
15
16 _start:
17     ; Ввод значения x
18     mov eax, msg_x
19     call sprint
20
21     mov ecx, x
22     mov edx, 10
23     call sread
24
25     ; Преобразование x из строки в число
26     mov eax, x
27     call atoi
28     mov [x], eax
29
30     ; Ввод значения a
31     mov eax, msg_a
```

Рисунок 3.18: Текст программы задания 2 из самостоятельной работы

Создаём исполняемый файл и проверяем его работу для значений  $x_1=1$ ,  $x_2=2$ ,  $a_1=2$ ,  $a_2=1$  из 7.6.



```
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$ nasm -f elf function.asm
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$ ld -m elf_i386 function.o -o function
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$ ./function
Введите x: 1
Введите a: 2
Результат f(x) = 5
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$ ./function
Введите x: 2
Введите a: 1
Результат f(x) = 1
gabduLLinaab@ubuntu:~/work/arch-pc/lab07$
```

Рисунок 3.19: Создание файла и проверка его работы

## 4 Выводы

В ходе лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.