

# **Отчет по лабораторной работе № 7**

**Дисциплина: архитектура компьютеров**

Казазаев Даниил Михайлович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задания Лабораторной работы</b>	<b>5</b>
<b>3</b>	<b>Задания Самостоятельной работы</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
<b>5</b>	<b>Выполнение самостоятельной работы</b>	<b>14</b>
5.1	Задание 1 . . . . .	14
5.2	Задание 2 . . . . .	18
<b>6</b>	<b>Вывод</b>	<b>22</b>

## Список иллюстраций

4.1	Создание файла lab7-1.asm . . . . .	7
4.2	Редактирование файла lab7-1.asm . . . . .	7
4.3	Трансляция и запуск файла lab7-1.asm . . . . .	8
4.4	Редактирование файла lab7-1.asm . . . . .	8
4.5	Трансляция и запуск файла lab6-1.asm . . . . .	9
4.6	Создание файла lab7-2.asm . . . . .	9
4.7	Редактирование файла lab7-2.asm . . . . .	10
4.8	Трансляция и запуск файла lab7-2.asm, с последующей проверкой для разных значений В . . . . .	11
4.9	Создание листинга . . . . .	11
4.10	Открытый листинг . . . . .	12
4.11	Выбранные строки . . . . .	12
4.12	Удаление одного из операндов . . . . .	13
4.13	Трансляция с получением файла листинга . . . . .	13
5.1	Создание файла task.asm . . . . .	14
5.2	Редактирую файл . . . . .	15
5.3	Трансляция и запуск файла . . . . .	15
5.4	Трансляция и запуск файла . . . . .	18
5.5	Редактирование файла task2.asm . . . . .	19
5.6	Трансляция файла task2.asm . . . . .	19
5.7	Результат работы программы task2.asm . . . . .	19

# 1 Цель работы

Целью работы является Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

## 2 Задания Лабораторной работы

1. Создать файл lab7-1.asm.
2. Отредактировать файл lab7-1.asm.
3. Создать исполняемый файл lab7-1.asm и запустить его.
4. Изменить текст программы.
5. Транслировать отредактированный файл lab7-1.asm в объектный файл и запустить его.
6. Создать файл lab7-2.asm.
7. Отредактировать файл lab7-2.asm.
8. Создать исполняемый файл lab7-2.asm и запустить его, проверив работу для разных значений В.
9. Создайте файл листинга для программы из файла lab7-2.asm.
10. Откройте файл листинга lab7-2.lst с помощью любого текстового редактора и изучите с его форматом и содержанием.
11. Подробно описать содержимое трёх строк.
12. Удалить один из операндов в файле lab7-2.asm, после чего выполнить трансляцию с получением файла листинга.

### 3 Задания Самостоятельной работы

1. Написать программу нахождения наименьшей из 3 целочисленных переменных  $a, b$  и  $c$ .
2. Написать программу, которая для введенных с клавиатуры значений  $x$  и  $y$  вычисляет значение заданной функции  $f(x, y)$  и выводит результат вычислений.

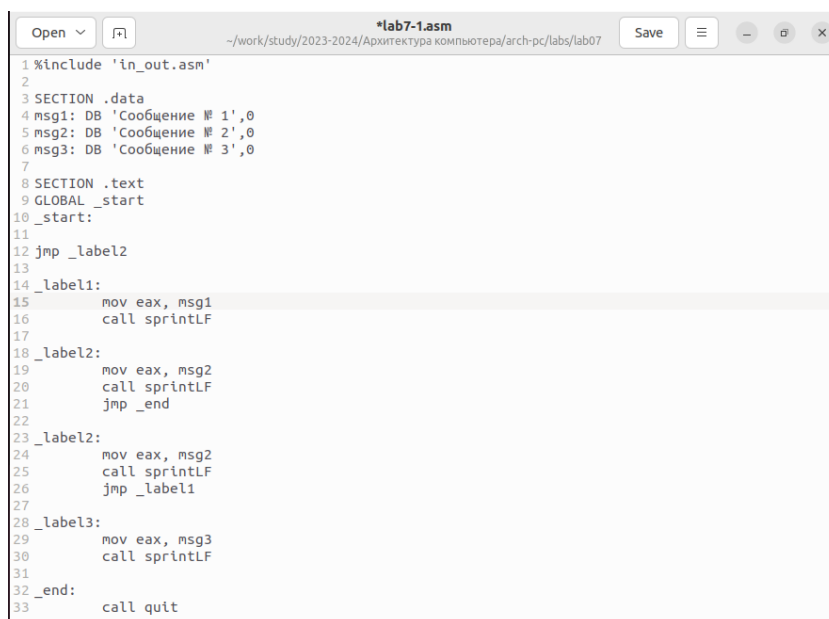
## 4 Выполнение лабораторной работы

Создаю файл lab7-1.asm. (рис. [4.1])

```
dmkazaev@Ubuntu:~$ cd ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab07
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07$ touch lab7-1.asm
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07$ ls
lab7-1.asm presentation report
```

Рис. 4.1: Создание файла lab7-1.asm

Редактирую файл lab7-1.asm. (рис. [4.2])



```
*lab7-1.asm
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15     mov eax, msg1
16     call sprintf
17
18 _label2:
19     mov eax, msg2
20     call sprintf
21     jmp _end
22
23 _label2:
24     mov eax, msg2
25     call sprintf
26     jmp _label1
27
28 _label3:
29     mov eax, msg3
30     call sprintf
31
32 _end:
33     call quit
```

Рис. 4.2: Редактирование файла lab7-1.asm

Транслирую файл lab7-1.asm в объектный файл, после чего запускаю его. (рис.

[4.3])

```
dmkazazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07$ nasm -f elf lab7-1.asm
dmkazazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmkazazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
```

Рис. 4.3: Трансляция и запуск файла lab7-1.asm

Немного редактирую файл lab7-1.asm. (рис. [4.4])

```
lab7-1.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label3
13
14 _label1:
15     mov eax, msg1
16     call sprintf
17     jmp _end
18
19 _label2:
20     mov eax, msg2
21     call sprintf
22     jmp _label1
23
24 _label3:
25     mov eax, msg3
26     call sprintf
27     jmp _label2
28
29 _end:
30     call quit
```

Рис. 4.4: Редактирование файла lab7-1.asm

Транслирую файл lab7-1.asm в объектный файл, после чего запускаю его. (рис. [4.5])



```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ nasm -f elf lab7-1.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ld -m elf_i386 -o lab7-1 lab7-1.o
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1

```

Рис. 4.5: Трансляция и запуск файла lab6-1.asm

Созаю файл lab7-2.asm. (рис. [4.6])

```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ touch lab7-2.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm presentation report

```

Рис. 4.6: Создание файла lab7-2.asm

Редактирую файл lab7-2.asm. (рис. [4.7])

lab7-1.asm	×	*lab7-2.asm
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		
8 section .bss		
9 max resb 10		
10 B resb 10		
11		
12		
13 section .text		
14 global _start		
15 _start:		
16 ; ----- Вывод сообщения 'Введите B: '		
17 mov eax,msg1		
18 call sprint		
19 ; ----- Ввод 'B'		
20 mov ecx,B		
21 mov edx,10		
22 call sread		
23 ; ----- Преобразование 'B' из символа в число		
24 mov eax,B		
25 call atoi ; Вызов подпрограммы перевода символа в число		
26 mov [B],eax ; запись преобразованного числа в 'B'		
27 ; ----- Записываем 'A' в переменную 'max'		
28 mov ecx,[A] ; 'ecx = A'		
29 mov [max],ecx ; 'max = A'		
30 ; ----- Сравниваем 'A' и 'C' (как символы)		
31 cmp ecx,[C] ; Сравниваем 'A' и 'C'		
32 jg check_B ; если 'A>C', то переход на метку 'check_B',		
33 mov ecx,[C] ; иначе 'ecx = C'		
34 mov [max],ecx ; 'max = C'		
35 ; ----- Преобразование 'max(A,C)' из символа в число		
36 check_B:		
37 mov eax,max		
38 call atoi ; Вызов подпрограммы перевода символа в число		
39 mov [max],eax ; запись преобразованного числа в 'max'		
40 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)		
41 mov ecx,[max]		
42 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'		
43 jg fin ; если 'max(A,C)>B', то переход на 'fin',		
44 mov ecx,[B] ; иначе 'ecx = B'		
45 mov [max],ecx		
46 ; ----- Вывод результата		

Рис. 4.7: Редактирование файла lab7-2.asm

Создать исполняемый файл lab7-2.asm и запустить его, проверив работу для разных значений B. (рис. [4.8])

```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ nasm -f elf lab7-2.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ld -m elf_i386 -o lab7-2 lab7-2.o
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ./lab7-2
Введите В: 3
Наибольшее число: 50
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ./lab7-2
Введите В: 51
Наибольшее число: 51
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ./lab7-2
Введите В: 45
Наибольшее число: 50

```

Рис. 4.8: Трансляция и запуск файла lab7-2.asm, с последующей проверкой для разных значений В

Создаю файл листинга для программы из файла lab7-2.asm с помощью команды `nasm -f elf -l lab7-2.lst lab7-2.asm`. (рис. [4.9])

```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ nasm -f elf -l lab7-2.lst lab7-2.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst  presentation
lab7-1      lab7-1.o    lab7-2.asm  lab7-2.o    report

```

Рис. 4.9: Создание листинга

Открываю листинг в текстовом редакторе, чтобы изучить содержимое. (рис. [4.10])

lab7-1.asm	lab7-2.asm	lab7-2.lst
1 1	%include 'in_out.asm'	
2 2	<1> ;----- slen -----	
3 3	<1> ; Функция вычисления длины сообщения	
4 4	<1> slen:	
5 5 00000000 6653	<1> push ebx	
6 6 00000002 6689C3	<1> mov ebx, eax	
7 7	<1>	
8 8	<1> nextchar:	
9 9 00000005 67803800	<1> cmp byte [eax], 0	
10 10 00000009 7404	<1> jz finished	
11 11 00000008 6640	<1> inc eax	
12 12 0000000D EBF6	<1> jmp nextchar	
13 13	<1>	
14 14	<1> finished:	
15 15 0000000F 6629D8	<1> sub eax, ebx	
16 16 00000012 665B	<1> pop ebx	
17 17 00000014 C3	<1> ret	
18 18	<1>	
19 19	<1>	
20 20	<1> ;----- sprint -----	
21 21	<1> ; Функция печати сообщения	
22 22	<1> ; входные данные: mov eax, <message>	
23 23	<1> sprint:	
24 24 00000015 6652	<1> push edx	
25 25 00000017 6651	<1> push ecx	
26 26 00000019 6653	<1> push ebx	
27 27 0000001B 6650	<1> push eax	
28 28 0000001D E8E0FF	<1> call slen	
29 29	<1>	
30 30 00000020 6689C2	<1> mov edx, eax	
31 31 00000023 6658	<1> pop eax	
32 32	<1>	
33 33 00000025 6689C1	<1> mov ecx, eax	
34 34 00000028 66BB01000000	<1> mov ebx, 1	

Рис. 4.10: Открытый листинг

Далее я буду описывать эти три строчки кода. (рис. [4.11])

10 10 00000006 7403	<1> jz finished
11 11 00000008 40	<1> inc eax
12 12 00000009 EBF8	<1> jmp nextchar

Рис. 4.11: Выбранные строчки

- 1) “10” - номер строки кода; “00000006” - адрес строки; “7403” - машинный код; “jz” - инструкция, выполняющая переход к метке, если флаг нуля установлен; “finished” - метка, к которой выполняется переход.
- 2) “11” - номер строки кода; “00000008” - адрес строки; “40” - машинный код; “inc” - команда, которая увеличивает число на единицу; “eax” - элемент, который повышается на единицу.
- 3) “12” - номер строки кода; “00000009” - адрес строки; “EBF8” - машинный код; “jmp” - инструкция выполняющая безусловный переход к метке; “nextchar” - метка, к которой выполняется переход.

Удаляю один из операндов в файле lab7-2.asm. (рис. [4.12])

```
40 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
41     mov ecx,[max]
42     cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
43     jg fin ; если 'max(A,C)>B', то переход на 'fin',
44     mov ecx,[B] ; иначе 'ecx = B'
45     mov [max],ecx
```

Рис. 4.12: Удаление одного из операндов

Выполняю трансляцию lab7-2.asm с получением файла листинга. (рис. [4.13])

```
dnkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:41: error: invalid combination of opcode and operands
```

Рис. 4.13: Трансляция с получением файла листинга

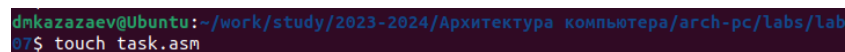
Выполнение команды `nasm -f elf -l lab7-2.lst lab7-2.asm` не создало ни-одного файла, так как инструкция “mov” не может работать имея только один операнд.

## 5 Выполнение самостоятельной работы

После выполнения прошлой лабораторной работы я получил вариант 8.

### 5.1 Задание 1

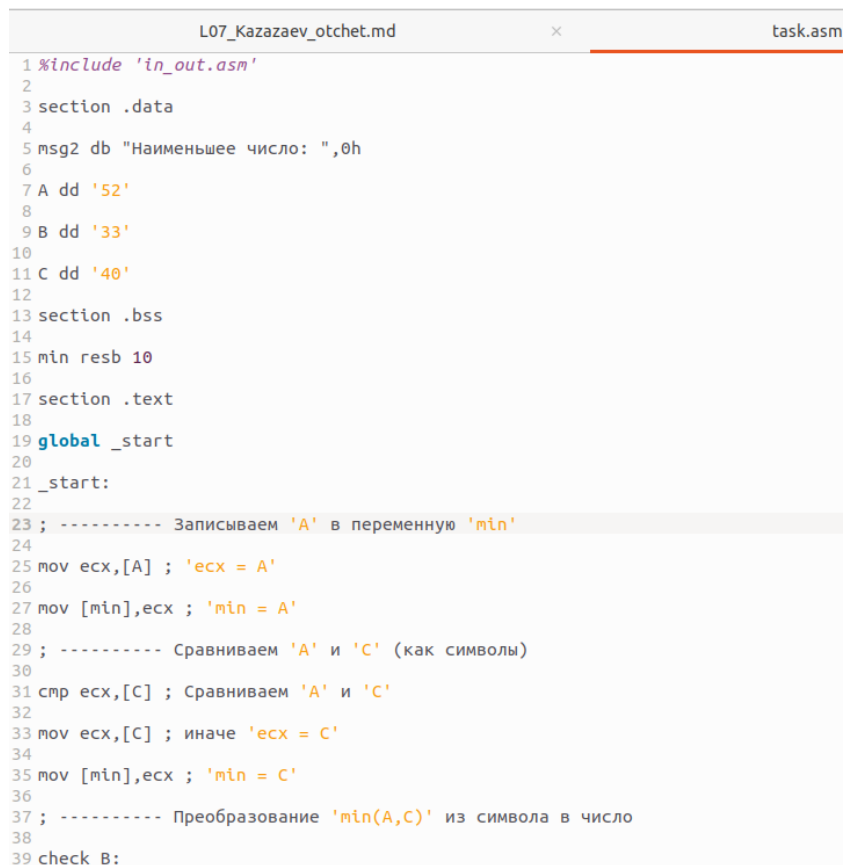
Создаю файл task.asm, в котором буду выполнять задание. (рис. [5.1])



```
dmkazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab  
07$ touch task.asm
```

Рис. 5.1: Созадние файла task.asm

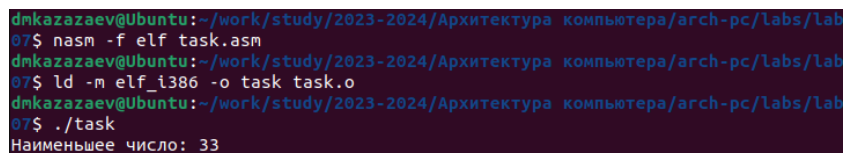
Редактирую файл task.asm. (рис. [5.2])



```
L07_Kazazaev_otchet.md x task.asm
1 %include 'in_out.asm'
2
3 section .data
4
5 msg2 db "Наименьшее число: ",0h
6
7 A dd '52'
8
9 B dd '33'
10
11 C dd '40'
12
13 section .bss
14
15 min resb 10
16
17 section .text
18
19 global _start
20
21 _start:
22
23 ; ----- Записываем 'A' в переменную 'min'
24
25 mov ecx,[A] ; 'ecx = A'
26
27 mov [min],ecx ; 'min = A'
28
29 ; ----- Сравниваем 'A' и 'C' (как символы)
30
31 cmp ecx,[C] ; Сравниваем 'A' и 'C'
32
33 mov ecx,[C] ; иначе 'ecx = C'
34
35 mov [min],ecx ; 'min = C'
36
37 ; ----- Преобразование 'min(A,C)' из символа в число
38
39 check_B:
```

Рис. 5.2: Редактирую файл

Транслирую файл task.asm в объектный файл, после чего запускаю его. (рис. [5.3])



```
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ nasm -f elf task.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ld -m elf_i386 -o task task.o
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab
07$ ./task
Наименьшее число: 33
```

Рис. 5.3: Трансляция и запуск файла

Программа работает корректно.

### Листинг task.asm

```
%include 'in_out.asm'
```

```
section .data
```

```
msg2 db "Наименьшее число: ",0h
```

```
A dd '52'
```

```
B dd '33'
```

```
C dd '40'
```

```
section .bss
```

```
min resb 10
```

```
section .text
```

```
global _start
```

```
_start:
```

```
; ----- 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'

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)

cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'

jg 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: '
```

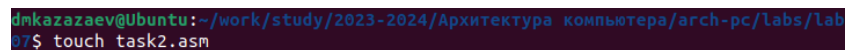
```
mov ecx,[min]
```

```
call iprintLF ; Вывод 'min(A,B,C)'
```

```
call quit ; Выход
```

## 5.2 Задание 2

Создаю файл task2.asm. (рис. [5.4])



```
dmkazaev@Ubuntu: ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab  
07$ touch task2.asm
```

Рис. 5.4: Трансляция и запуск файла

Открываю его и начинаю писать программу для вычисления заданной функции  $f(x)$ . Так как у меня восьмой вариант, мой вид функции:  $3 \cdot a$ , если  $a < 3$  и  $x + 1$ , если  $a \geq 3$ . (рис. [5.5])

```

1 %include 'in_out.asm'
2
3 section .data
4
5     msgx : db "Введите x:",0
6     msga : db "Введите a:",0
7     msgres : db "Результат:",0
8
9 section .bss
10
11     x: resb 80
12     a: resb 80
13
14 section .text
15 global _start
16 _start:
17     mov eax,msga
18     call sprint
19     mov ecx, a
20     mov edx, 80
21     call sread
22     mov eax, a
23     call atoi
24     cmp eax, 3
25     jl _funca
26
27     mov eax,msgx
28     call sprint
29     mov ecx,x
30     mov edx,80
31     call sread
32     mov eax,x
33     call atoi
34     jge _funcx
35
36 _funcx:
37     add eax, 1

```

Рис. 5.5: Редактирование файла task2.asm

Транслирую файл task2.asm в объектный файл. (рис. [5.6])

```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab0
$ nasm -f elf task2.asm
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab0
$ ld -m elf_i386 -o task2 task2.o

```

Рис. 5.6: Трансляция файла task2.asm

Проверяю исправность работы программы. (рис. [5.7])

```

dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab0
$ ./task2
Введите a:4
Введите x:1
Результат:2
dmkazazaev@Ubuntu:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab0
$ ./task2
Введите a:2
Результат:6

```

Рис. 5.7: Результат работы программы task2.asm

Пришлось изменить порядок ввода значений x и a, чтобы программа работала корректно

### **Листинг task2.asm**

```
%include 'in_out.asm'
```

```
section .data
```

```
msgx : db "Введите x:",0  
msga : db "Введите a:",0  
msgres : db "Результат:",0
```

```
section .bss
```

```
x: resb 80  
a: resb 80
```

```
section .text
```

```
global _start
```

```
_start:
```

```
    mov eax,msga  
    call sprint  
    mov ecx, a  
    mov edx, 80  
    call sread  
    mov eax, a  
    call atoi  
    cmp eax, 3  
    jl _funca
```

```
mov eax,msgx
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
jge _funcx
```

```
_funcx:
    add eax, 1
    jmp _fin
```

```
_funca:
    mov edx,3
    mul edx
    jmp _fin
```

```
_fin:
    mov ecx, eax
    mov eax, msgres
    call sprint
    mov eax,ecx
    call iprintLF
    call quit
```

## 6 Вывод

При выполнении данной лабораторной работы я освоил команды условного и безусловного переходов, приобрел навыки написания программ с использованием переходов и ознакомился с назначением и структурой файла листинга.