

Отчёт по лабораторной работе №7

Дисциплина: архитектура компьютера

Учаева Алёна Сергеевна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	7
4.1	Реализация переходов в NASM	7
4.2	Изучение структуры файлы листинга	12
4.3	Задания для самостоятельной работы	14
5	Выводы	21
	Список литературы	22

Список иллюстраций

4.1	Создание каталога	7
4.2	Редактирование файла	8
4.3	Запуск исполняемого файла	8
4.4	Редактирование файла	9
4.5	Запуск исполняемого файла	9
4.6	Редактирование файла	10
4.7	Запуск исполняемого файла	10
4.8	Редактирование файла	11
4.9	Запуск исполняемого файла	12
4.10	Создание файла	13
4.11	Редактирование файла	14
4.12	Трансляция файла	14
4.13	Редактирование файла	15
4.14	Запуск исполняемого файла	15
4.15	Создание программы	18
4.16	Запуск исполняемого файла	18

1 Цель работы

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

2 Задание

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

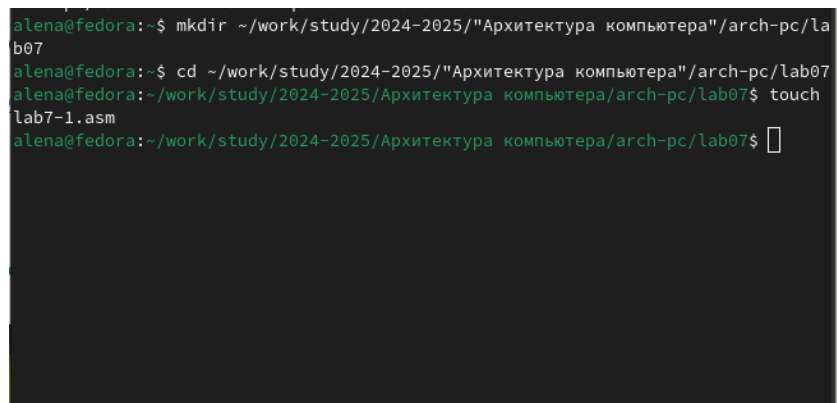
3 Теоретическое введение

Для реализации ветвлений в ассемблере используются так называемые команды передачи управления или команды перехода. Можно выделить 2 типа переходов: • условный переход – выполнение или не выполнение перехода в определенную точку программы в зависимости от проверки условия. • безусловный переход – выполнение передачи управления в определенную точку программы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

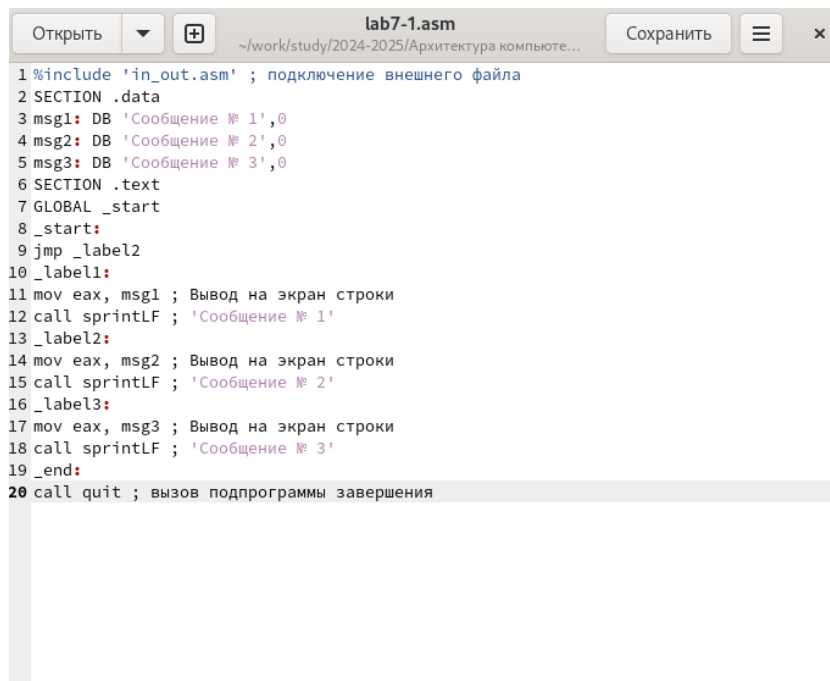
Создаю каталог для программ лабораторной работы №7 (рис. 4.1).



```
alena@fedora:~$ mkdir ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab07
alena@fedora:~$ cd ~/work/study/2024-2025/"Архитектура компьютера"/arch-pc/lab07
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.1: Создание каталога

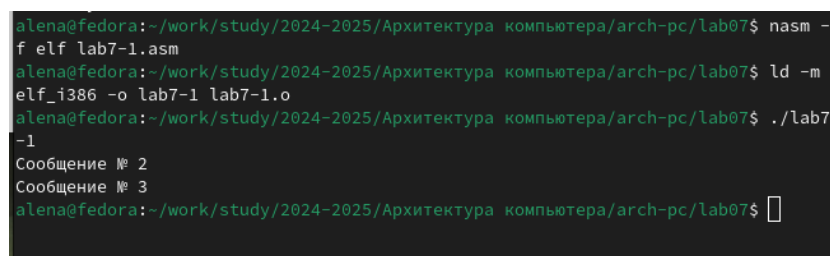
Ввожу в файл lab7-1.asm текст программы из листинга 7.1 (рис. 4.2).



```
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 _label2:
14 mov eax, msg2 ; Вывод на экран строки
15 call sprintf ; 'Сообщение № 2'
16 _label3:
17 mov eax, msg3 ; Вывод на экран строки
18 call sprintf ; 'Сообщение № 3'
19 _end:
20 call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Редактирование файла

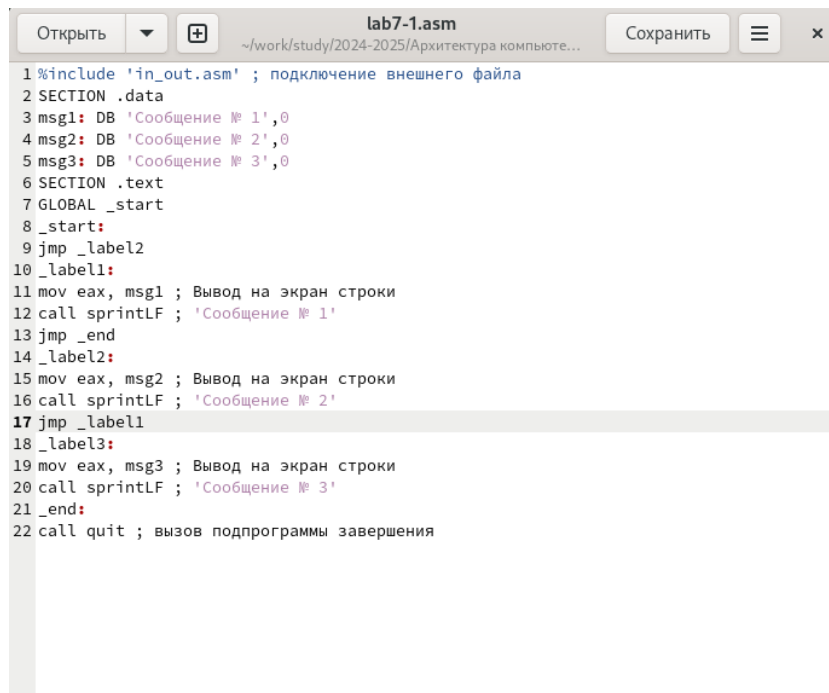
Создаю исполняемый файл и запускаю его (рис. 4.3).



```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
-1
Сообщение № 2
Сообщение № 3
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.3: Запуск исполняемого файла

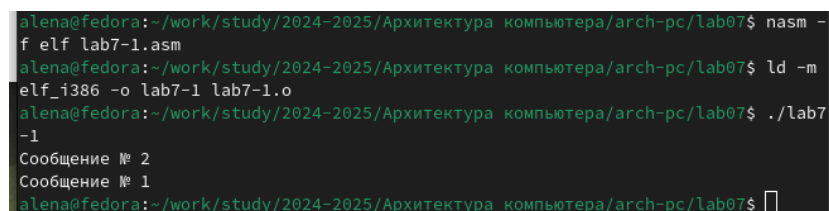
Редактирую программу, чтобы поменялся порядок выполнения функций (рис. 4.4).



```
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 sprintfLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg2 ; Вывод на экран строки
16 call sprintfLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintfLF ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Редактирование файла

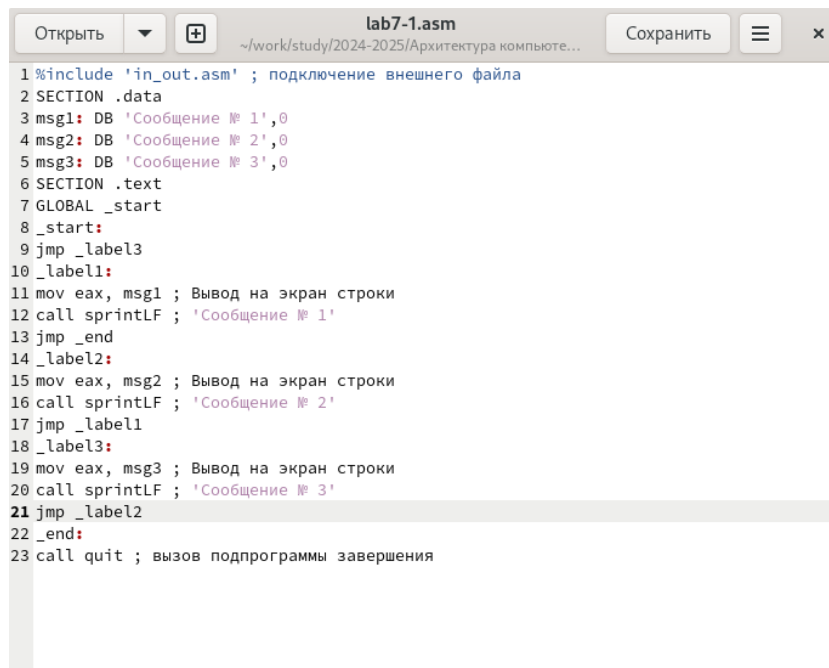
Создаю исполняемый файл и запускаю его (рис. 4.5).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.5: Запуск исполняемого файла

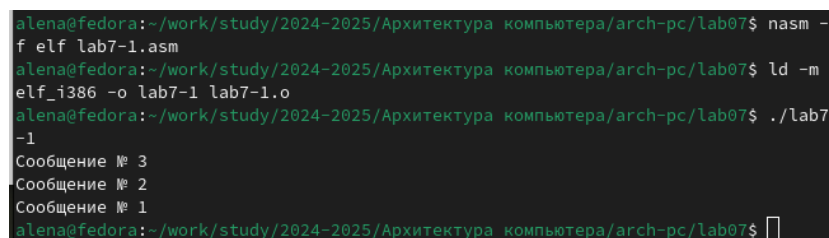
Редактирую текст программы, чтобы сообщения вывелись в обратном порядке (рис. 4.6).



```
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
```

Рис. 4.6: Редактирование файла

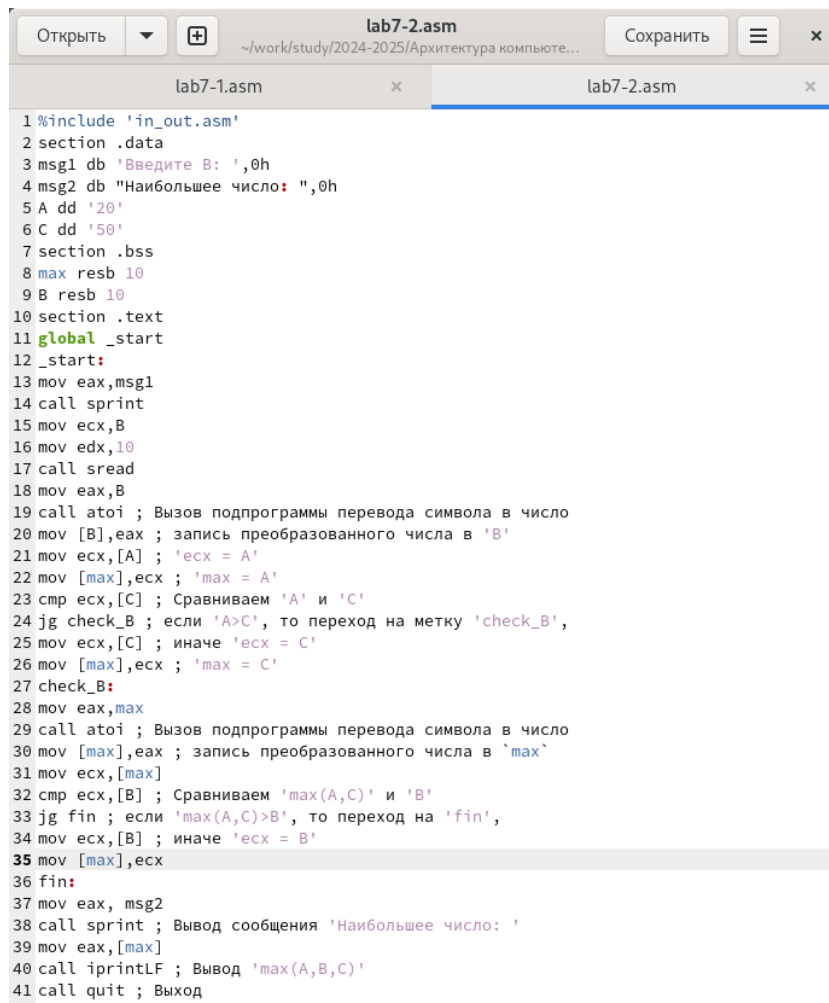
Создаю исполняемый файл и запускаю его (рис. 4.7).



```
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.7: Запуск исполняемого файла

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и ввожу в него текст программы из листинга 7.3 (рис. 4.8).



```
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 mov eax,msg1
14 call sprint
15 mov ecx,B
16 mov edx,10
17 call sread
18 mov eax,B
19 call atoi ; Вызов подпрограммы перевода символа в число
20 mov [B],eax ; запись преобразованного числа в 'B'
21 mov ecx,[A] ; 'ecx = A'
22 mov [max],ecx ; 'max = A'
23 cmp ecx,[C] ; Сравниваем 'A' и 'C'
24 jg check_B ; если 'A>C', то переход на метку 'check_B',
25 mov ecx,[C] ; иначе 'ecx = C'
26 mov [max],ecx ; 'max = C'
27 check_B:
28 mov eax,max
29 call atoi ; Вызов подпрограммы перевода символа в число
30 mov [max],eax ; запись преобразованного числа в 'max'
31 mov ecx,[max]
32 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
33 jg fin ; если 'max(A,C)>B', то переход на 'fin',
34 mov ecx,[B] ; иначе 'ecx = B'
35 mov [max],ecx
36 fin:
37 mov eax, msg2
38 call sprint ; Вывод сообщения 'Наибольшее число: '
39 mov eax,[max]
40 call iprintLF ; Вывод 'max(A,B,C)'
41 call quit ; Выход
```

Рис. 4.8: Редактирование файла

Создаю исполняемый файл и запускаю его. Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 4.9).

```

alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch
lab7-2.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -
f elf lab7-2.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m
elf_i386 -o lab7-2 lab7-2.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7
-2
Введите В: 30
Наибольшее число: 50
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7
-2
Введите В: 10
Наибольшее число: 50
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7
-2
Введите В: 45
Наибольшее число: 50
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7
-2
Введите В: 70
Наибольшее число: 70
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ 

```

Рис. 4.9: Запуск исполняемого файла

4.2 Изучение структуры файлы листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора (рис. 4.10).

```

lab7-2.lst
~/work/study/2024-2025/Архитектура компьюте...
lab7-1.asm lab7-2.asm lab7-2.lst
1 1 %include 'in_out.asm'
2 1 <1> ;----- slen
3 2 <1> ; Функция вычисления длины сообщения
4 3 <1> slen:
5 4 00000000 53 <1> push ebx
6 5 00000001 89C3 <1> mov ebx, eax
7 6 <1>
8 7 <1> nextchar:
9 8 00000003 803800 <1> cmp byte [eax], 0
10 9 00000006 7403 <1> jz finished
11 10 00000008 40 <1> inc eax
12 11 00000009 EBF8 <1> jmp nextchar
13 12 <1>
14 13 <1> finished:
15 14 0000000B 29D8 <1> sub eax, ebx
16 15 0000000D 5B <1> pop ebx
17 16 0000000E C3 <1> ret
18 17 <1>
19 18 <1>
20 19 <1> ;----- sprint
21 20 <1> ; Функция печати сообщения
22 21 <1> ; входные данные: mov eax, <message>
23 22 <1> sprint:
24 23 0000000F 52 <1> push edx
25 24 00000010 51 <1> push ecx
26 25 00000011 53 <1> push ebx
27 26 00000012 50 <1> push eax
28 27 00000013 E8E8FFFFFF <1> call slen
29 28 <1>
30 29 00000018 89C2 <1> mov edx, eax
31 30 0000001A 58 <1> pop eax
32 31 <1>
33 32 0000001B 89C1 <1> mov ecx, eax
34 33 0000001D BB01000000 <1> mov ebx, 1
35 34 00000022 B804000000 <1> mov eax, 4
36 35 00000027 CD80 <1> int 80h
37 36 <1>
38 37 00000029 5B <1> pop ebx
39 38 0000002A 59 <1> pop ecx
40 39 0000002B 5A <1> pop edx
41 40 0000002C C3 <1> ret
42 41 <1>
43 42 <1>

```

Рис. 4.10: Создание файла

Строка 5: 5 00000001 89C3 Данная команда перемещает данные из одного регистра в другой, сохраняет начальный адрес строки в регистре ebx. Строка 8: 00000003 803800 Данная команда проверяет, является ли текущий символ нулевым (конец строки). Строка 27: 00000013 E8E8FFFFFF Данная команда вызывает функцию slen, которая вычисляет длину строки.

Далее удаляю один операнд из случайной инструкции (рис. 4.11).

```

14 call sprint
15 mov ecx,B
16 mov edx,10
17 call sread
18 mov eax,B
19 call atoi ; Вызов подпрограммы перевода символа в число
20 mov eax ; запись преобразованного числа в 'B'
21 mov ecx,[A] ; 'ecx = A'
22 mov [max],ecx ; 'max = A'

```

Рис. 4.11: Редактирование файла

Выполняя трансляцию с получением файла листинга, показывает ошибку. Никакие выходные при этом не создаются (рис. 4.12).

```

alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -
f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:20: error: invalid combination of opcode and operands
alena@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$

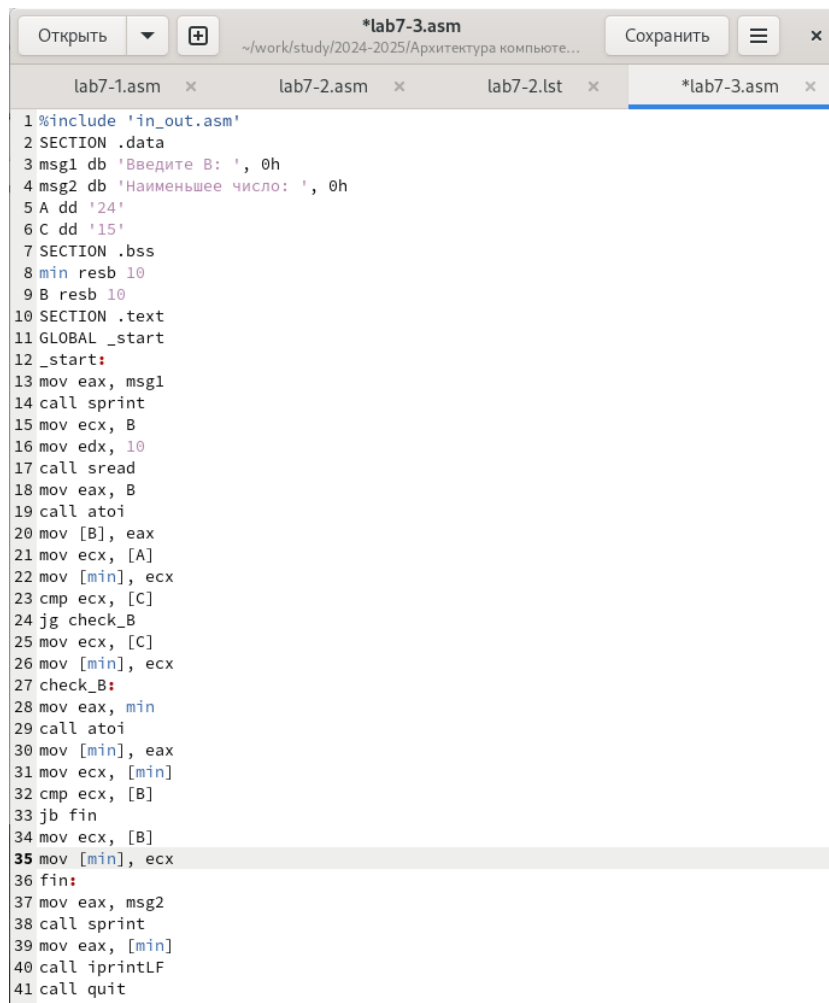
```

Рис. 4.12: Трансляция файла

4.3 Задания для самостоятельной работы

Вариант №9(мой вариант из лабораторной работы №6):

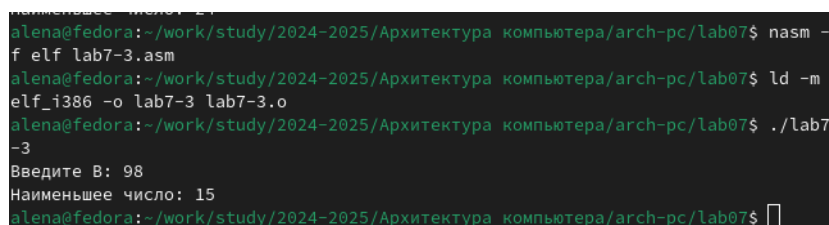
Редактирую программу для нахождения наименьшей из 3 целочисленных переменных (рис. 4.13).



```
Открыть ▼ + *lab7-3.asm ~/work/study/2024-2025/Архитектура компьюте... Сохранить ≡ ×
lab7-1.asm × lab7-2.asm × lab7-2.lst × *lab7-3.asm ×
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите B: ', 0h
4 msg2 db 'Наименьшее число: ', 0h
5 A dd '24'
6 C dd '15'
7 SECTION .bss
8 min resb 10
9 B resb 10
10 SECTION .text
11 GLOBAL _start
12 _start:
13 mov eax, msg1
14 call sprint
15 mov ecx, B
16 mov edx, 10
17 call sread
18 mov eax, B
19 call atoi
20 mov [B], eax
21 mov ecx, [A]
22 mov [min], ecx
23 cmp ecx, [C]
24 jg check_B
25 mov ecx, [C]
26 mov [min], ecx
27 check_B:
28 mov eax, min
29 call atoi
30 mov [min], eax
31 mov ecx, [min]
32 cmp ecx, [B]
33 jb fin
34 mov ecx, [B]
35 mov [min], ecx
36 fin:
37 mov eax, msg2
38 call sprint
39 mov eax, [min]
40 call iprintLF
41 call quit
```

Рис. 4.13: Редактирование файла

Создаю исполняемый файл и запускаю его, программа работает корректно (рис. 4.14).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-3.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-3
Введите B: 98
Наименьшее число: 15
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.14: Запуск исполняемого файла

```
%include 'in_out.asm'
SECTION .data
```

```

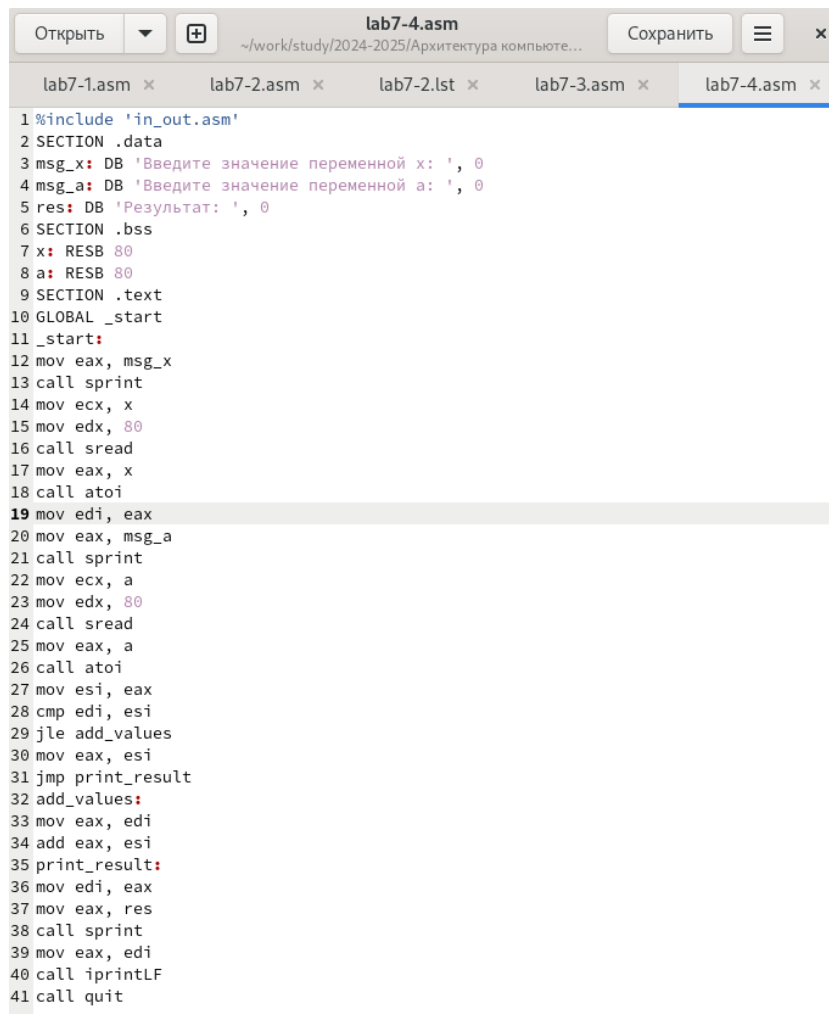
msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h
A dd '24'
C dd '15'
SECTION .bss
min resb 10
B resb 10
SECTION .text
GLOBAL _start
_start:
mov eax, msg1
call sprint
mov ecx, B
mov edx, 10
call sread
mov eax, B
call atoi
mov [B], eax
mov ecx, [A]
mov [min], ecx
cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx
check_B:
mov eax, min
call atoi
mov [min], eax
mov ecx, [min]

```



```
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx
fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit
```

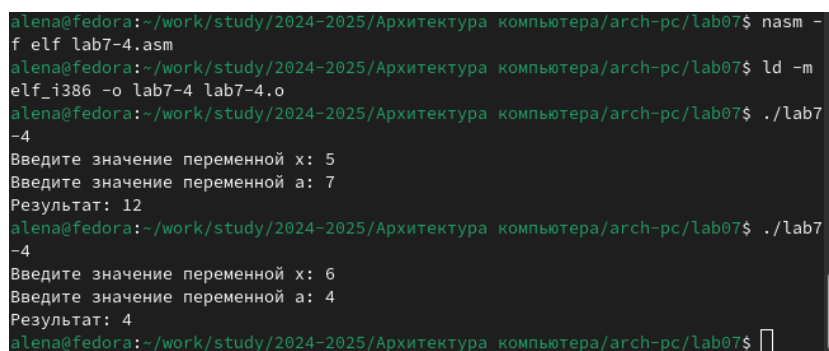
2. Пишу программу, которая вычисляет значение заданной функции и выводит результат вычислений.(рис. 4.15).



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg_x: DB 'Введите значение переменной x: ', 0
4 msg_a: DB 'Введите значение переменной a: ', 0
5 res: DB 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 a: RESB 80
9 SECTION .text
10 GLOBAL _start
11 _start:
12 mov eax, msg_x
13 call sprint
14 mov ecx, x
15 mov edx, 80
16 call sread
17 mov eax, x
18 call atoi
19 mov edi, eax
20 mov eax, msg_a
21 call sprint
22 mov ecx, a
23 mov edx, 80
24 call sread
25 mov eax, a
26 call atoi
27 mov esi, eax
28 cmp edi, esi
29 jle add_values
30 mov eax, esi
31 jmp print_result
32 add_values:
33 mov eax, edi
34 add eax, esi
35 print_result:
36 mov edi, eax
37 mov eax, res
38 call sprint
39 mov eax, edi
40 call iprintLF
41 call quit
```

Рис. 4.15: Создание программы

Создаю исполняемый файл и запускаю его, программа работает корректно (рис. 4.16).



```
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-4.asm
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
alena@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 4.16: Запуск исполняемого файла

```

#include 'in_out.asm'

SECTION .data
msg_x: DB 'Введите значение переменной x: ', 0
msg_a: DB 'Введите значение переменной a: ', 0
res: DB 'Результат: ', 0

SECTION .bss
x: RESB 80
a: RESB 80

SECTION .text
GLOBAL _start
_start:
mov eax, msg_x
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax
mov eax, msg_a
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax
cmp edi, esi
jle add_values

```

```
mov eax, esi
jmp print_result
add_values:
mov eax, edi
add eax, esi
print_result:
mov edi, eax
mov eax, res
call sprint
mov eax, edi
call iprintLF
call quit
```

5 Выводы

При выполнении данной лабораторной работы я изучила команды условного и безусловного переходов. Приобрела навыки написания программ с использованием переходов. Ознакомилась с назначением и структурой файла листинга.

Список литературы

1. Архитектура ЭВМ