

# **Отчет по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Краснова Камилла Геннадьевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Реализация переходов в NASM . . . . .	8
4.2	Изучение структуры файла листинга . . . . .	13
4.3	Выполнение заданий для самостоятельной работы . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>20</b>
<b>6</b>	<b>Список литературы</b>	<b>21</b>

# Список иллюстраций

4.1	Создание директории . . . . .	8
4.2	Редактирование файла . . . . .	9
4.3	Запуск исполняемого файла . . . . .	9
4.4	Запуск исполняемого файла . . . . .	11
4.5	Редактирование файла . . . . .	11
4.6	Запуск исполняемого файла . . . . .	12
4.7	Создание файла . . . . .	12
4.8	Редактирование файла . . . . .	12
4.9	Запуск исполняемого файла . . . . .	13
4.10	Создание файла . . . . .	13
4.11	Редактирование файла . . . . .	14
4.12	Трансляция с получением файла листинга . . . . .	14
4.13	Создание файла . . . . .	14
4.14	Редактирование файла . . . . .	15
4.15	Запуск исполняемого файла . . . . .	15
4.16	Создание файла . . . . .	15
4.17	Редактирование файла . . . . .	16
4.18	Запуск исполняемого файла . . . . .	16

## **Список таблиц**

# 1 Цель работы

Цель данной лабораторной работы - изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, знакомство с назначением и структурой файла листинга.

## **2 Задание**

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

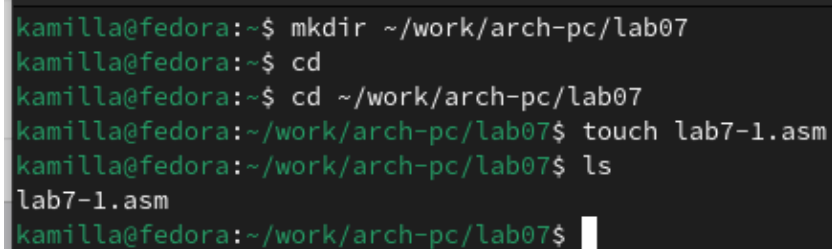
### 3 Теоретическое введение

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. Безусловный переход выполняется инструкцией `jmp` (от англ. `jump` – прыжок), которая включает в себя адрес перехода, куда следует передать управление: `jmp Адрес перехода` может быть либо меткой, либо адресом области памяти, в которую предварительно помещен указатель перехода. Кроме того, в качестве операнда можно использовать имя регистра, в таком случае переход будет осуществляться по адресу, хранящемуся в этом регистре. Как отмечалось выше, для условного перехода необходима проверка какого-либо условия. В ассемблере команды условного перехода вычисляют условие перехода анализируя флаги из регистра флагов. Флаг – это бит, принимающий значение 1 («флаг установлен»), если выполнено некоторое условие, и значение 0 («флаг сброшен») в противном случае. Флаги работают независимо друг от друга, и лишь для удобства они помещены в единый регистр — регистр флагов, отражающий текущее состояние процессора.

## 4 Выполнение лабораторной работы

### 4.1 Реализация переходов в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №7. Перехожу в созданный каталог с помощью утилиты `cd` и с помощью утилиты `touch` создаю файл `lab7-1.asm`. (рис. 4.1).

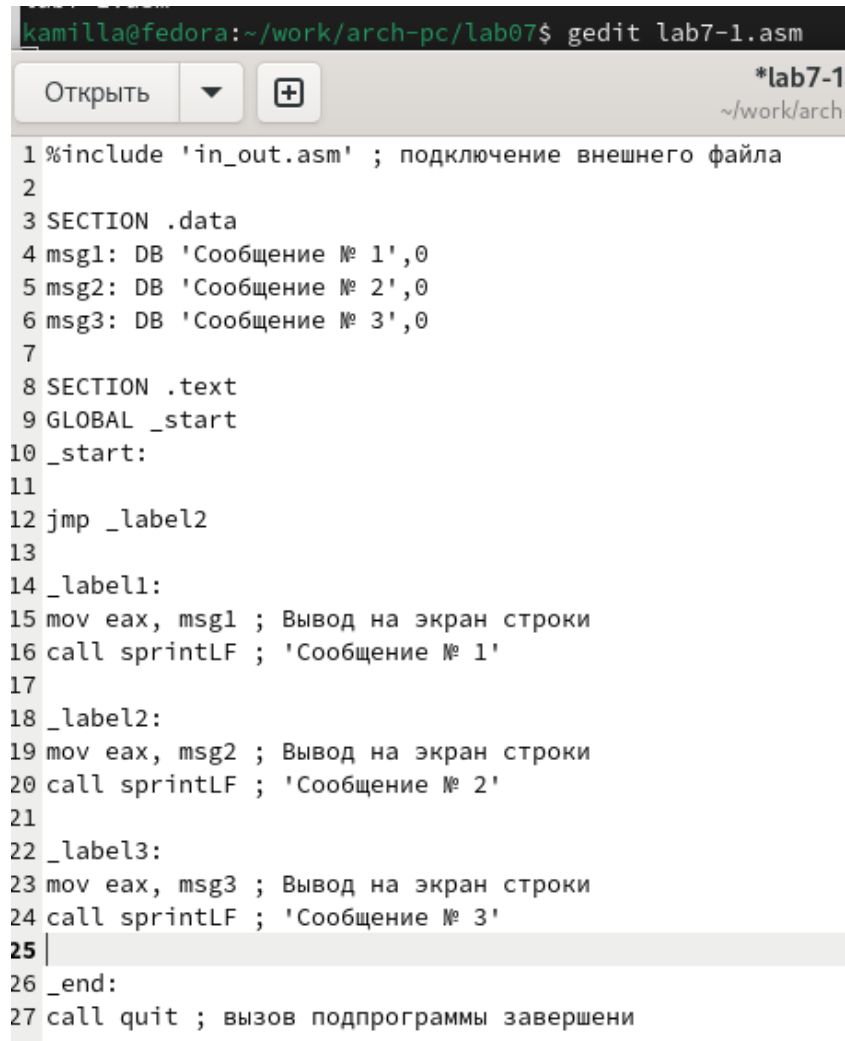


```
kamilla@fedora:~$ mkdir ~/work/arch-pc/lab07
kamilla@fedora:~$ cd
kamilla@fedora:~$ cd ~/work/arch-pc/lab07
kamilla@fedora:~/work/arch-pc/lab07$ touch lab7-1.asm
kamilla@fedora:~/work/arch-pc/lab07$ ls
lab7-1.asm
kamilla@fedora:~/work/arch-pc/lab07$
```

Рис. 4.1: Создание директории

Открываю созданный файл `lab7-1.asm`, вставляю в него программу с использованием инструкции `jmp` (рис. 4.2).





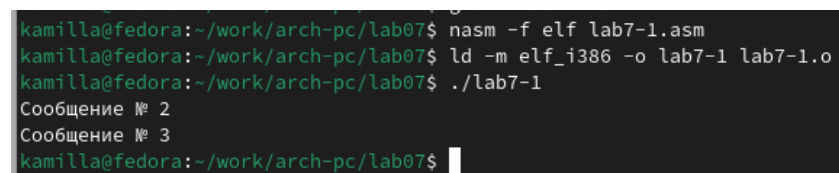
```
kamilla@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm

Открыть ▼ + *lab7-1
~/work/arch

1 %include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 1'
17
18 _label2:
19 mov eax, msg2 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 2'
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintLF ; 'Сообщение № 3'
25 |
26 _end:
27 call quit ; вызов подпрограммы завершени
```

Рис. 4.2: Редактирование файла

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



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

Рис. 4.3: Запуск исполняемого файла

Изменяю текст программы (рис. ??).

```

1 %include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintf ; 'Сообщение № 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call sprintf ; 'Сообщение № 3'
27
28 _end:
29 call quit ; вызов подпрограммы завершени

```

}

#fig:006=4 width=70% }

Создаю новый исполняемый файл программы и запускаю его (рис. 4.4).

```

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

```

Рис. 4.4: Запуск исполняемого файла

Изменяю текст программы, чтобы она выводила сначала сообщение №3, затем сообщение №2 и затем сообщение №1 (рис. 4.5).

```

1 %include 'in_out.asm' ; подключение внешнего файла
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 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 1'
17 jmp _end
18
19 _label2:
20 mov eax, msg2 ; Вывод на экран строки
21 call sprintLF ; 'Сообщение № 2'
22 jmp _label1
23
24 _label3:
25 mov eax, msg3 ; Вывод на экран строки
26 call sprintLF ; 'Сообщение № 3'
27 jmp _label2
28
29 _end:
30 call quit ; вызов подпрограммы завершени

```

Рис. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. 4.6).

```

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

```

Рис. 4.6: Запуск исполняемого файла

Создаю файл lab7-2 в каталоге ~/work/arch-pc/lab07 (рис. 4.7).

```

kamilla@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
kamilla@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
kamilla@fedora:~/work/arch-pc/lab07$

```

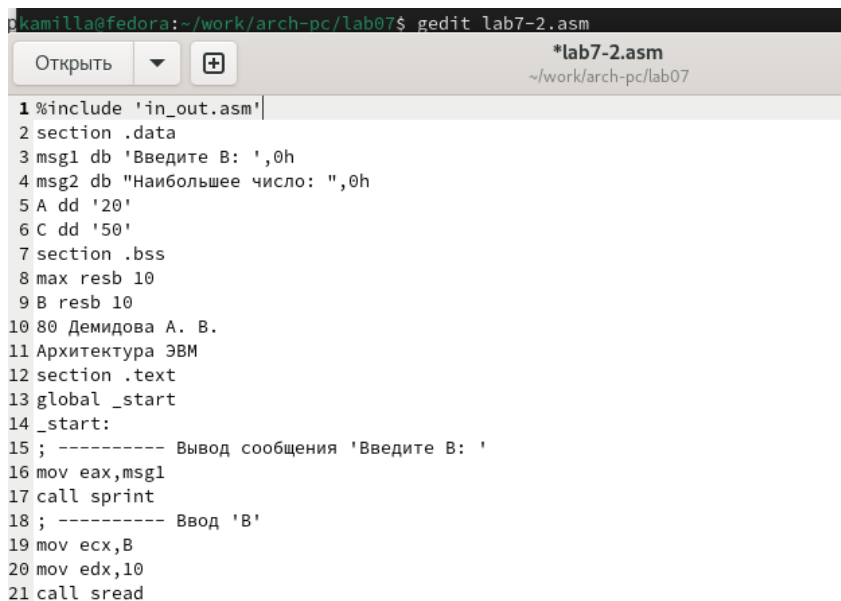
Рис. 4.7: Создание файла

Открываю созданный файл lab7-2.asm, вставляю в него программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. (рис. 4.8).

```

kamilla@fedora:~/work/arch-pc/lab07$ gedit lab7-2.asm

```



```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите В: ',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 80 Демидова А. В.
11 Архитектура ЭВМ
12 section .text
13 global _start
14 _start:
15 ; ----- Вывод сообщения 'Введите В: '
16 mov eax,msg1
17 call sprint
18 ; ----- Ввод 'В'
19 mov ecx,B
20 mov edx,10
21 call sread

```

Рис. 4.8: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.9). Проверяю работу для разных значений В. Программа работает верно.

```

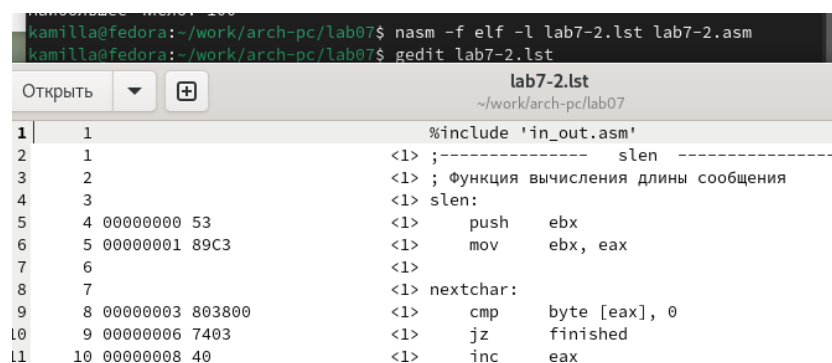
kamilla@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
kamilla@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
kamilla@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 100
Наибольшее число: 100
kamilla@fedora:~/work/arch-pc/lab07$

```

Рис. 4.9: Запуск исполняемого файла

## 4.2 Изучение структуры файла листинга

Создаю файл листинга для программы из файла lab7-2.asm, с помощью ключа -l и открываю файл листинга с помощью gedit (рис. 4.10).



```

kamilla@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
kamilla@fedora:~/work/arch-pc/lab07$ gedit lab7-2.lst

```

Line	Address	Code
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

Рис. 4.10: Создание файла

Первое - номер строки файла листинга, важно понимать, что он может не совпадать с номером строки в файле с исходным текстом. Дальше идет адрес - смещение машинного кода от начала текущего сегмента. Следом идет машинный код, который представляет собой ассемблированную исходную строку в виде шестнадцатиричной последовательности. И исходный текст программы - строка исходной программы вместе с комментариями.

Открываю файл с программой lab7-2.asm и удаляю один операнд (рис. 4.11).

```

kamilla@fedora:~/work/arch-pc/lab07$ gedit lab7-2.asm
Открыть *lab7-2.asm ~/work/arch-pc/lab07
14 ; ----- Вывод сообщения 'Введите B: '
15 mov eax,msg1
16 call sprint
17 ; ----- Ввод 'B'
18 mov ecx,B
19 mov edx,10
20 call sread
21 ; ----- Преобразование 'B' из символа в число
22 mov eax,B
23 call atoi ; Вызов подпрограммы перевода символа в число
24 mov [B],eax ; запись преобразованного числа в 'B'
25 ; ----- Записываем 'A' в переменную 'max'
26 mov ecx,[A] ; 'ecx = A'
27 mov [max],ecx ; 'max = A'
28 ; ----- Сравниваем 'A' и 'C' (как символы)
29 cmp ecx,[C] ; Сравниваем 'A' и 'C'
30 jg check_B ; если 'A>C', то переход на метку 'check_B',
31 mov ecx,[C] ; иначе 'ecx = C'
32 mov [max],ecx ; 'max = C'
33 ; ----- Преобразование 'max(A,C)' из символа в число
34 check_B:
35 mov eax,|

```

Рис. 4.11: Редактирование файла

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции. Не создаются входные данные (рис. 4.12).

```

kamilla@fedora:~/work/arch-pc/lab07$ gedit lab7-2.asm
kamilla@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:35: error: invalid combination of opcode and operands
kamilla@fedora:~/work/arch-pc/lab07$

```

Рис. 4.12: Трансляция с получением файла листинга

## 4.3 Выполнение заданий для самостоятельной работы

1. Создаю файл lab7-3.asm с помощью утилиты touch (рис. 4.13).

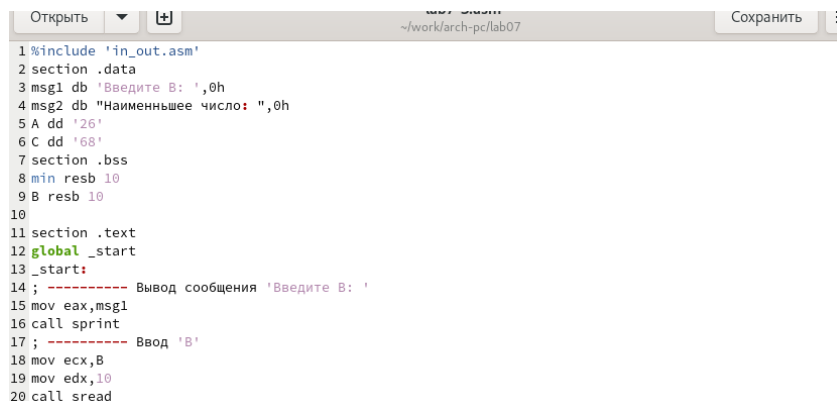
```

kamilla@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
kamilla@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1.asm lab7-2 lab7-2.lst
lab7-1 lab7-1.o lab7-2.asm lab7-3.asm
kamilla@fedora:~/work/arch-pc/lab07$

```

Рис. 4.13: Создание файла

Ввожу в созданный файл программу нахождения наименьшей из 3 целочисленных переменных `a`, `b` и `c`. (рис. 4.14).



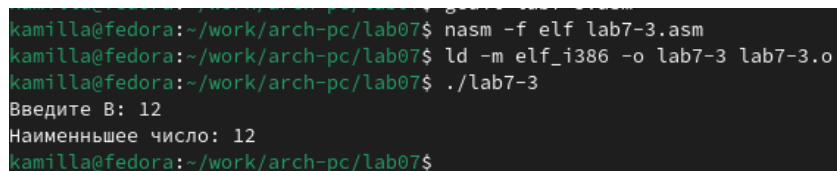
```

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наименьшее число: ",0h
5 A dd '26'
6 C dd '68'
7 section .bss
8 min resb 10
9 B resb 10
10
11 section .text
12 global _start
13 _start:
14 ; ----- Вывод сообщения 'Введите B: '
15 mov eax,msg1
16 call sprint
17 ; ----- Ввод 'B'
18 mov ecx,B
19 mov edx,10
20 call sread

```

Рис. 4.14: Редактирование файла

Создаю и запускаю новый исполняемый файл. Мой вариант - 17, ввожу соответствующие значения (рис. 4.15).



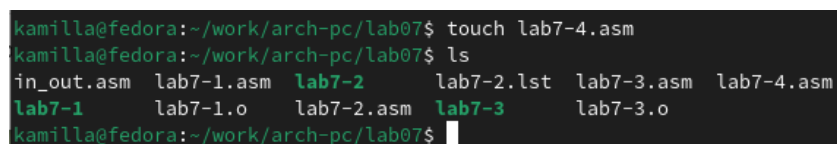
```

kamilla@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
kamilla@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
kamilla@fedora:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 12
Наименьшее число: 12
kamilla@fedora:~/work/arch-pc/lab07$

```

Рис. 4.15: Запуск исполняемого файла

2. Создаю файл lab7-4.asm с помощью утилиты touch (рис. 4.16).



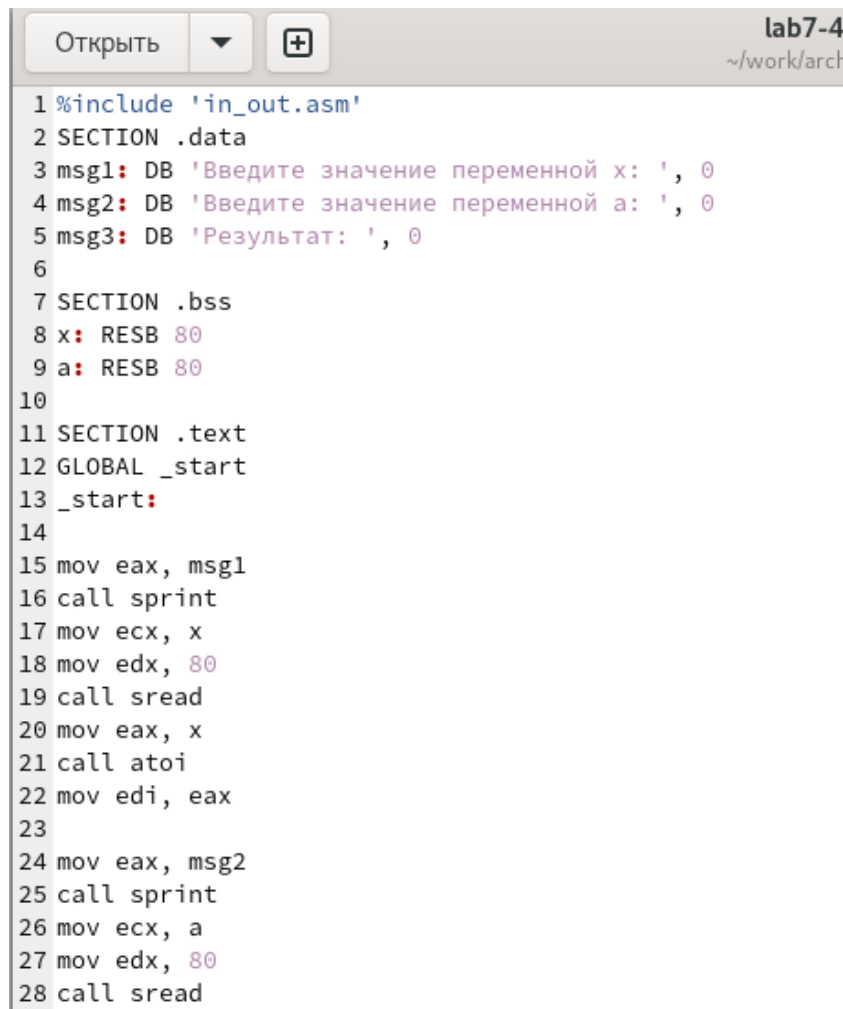
```

kamilla@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
kamilla@fedora:~/work/arch-pc/lab07$ ls
in_out.asm  lab7-1.asm  lab7-2      lab7-2.lst  lab7-3.asm  lab7-4.asm
lab7-1      lab7-1.o    lab7-2.asm  lab7-3      lab7-3.o
kamilla@fedora:~/work/arch-pc/lab07$

```

Рис. 4.16: Создание файла

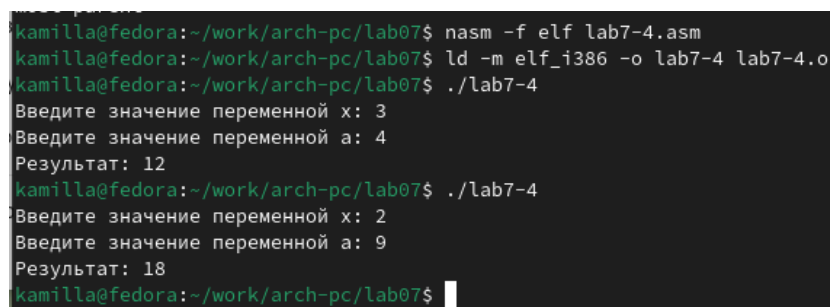
Ввожу в созданный файл программу которая для введенных с клавиатуры значений `a` и `b` вычисляет значение заданной функции `f(a,b)` и выводит результат вычислений (рис. 4.17).



```
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: ', 0
6
7 SECTION .bss
8 x: RESB 80
9 a: RESB 80
10
11 SECTION .text
12 GLOBAL _start
13 _start:
14
15 mov eax, msg1
16 call sprint
17 mov ecx, x
18 mov edx, 80
19 call sread
20 mov eax, x
21 call atoi
22 mov edi, eax
23
24 mov eax, msg2
25 call sprint
26 mov ecx, a
27 mov edx, 80
28 call sread
```

Рис. 4.17: Редактирование файла

Создаю и запускаю исполняемый файл (рис. 4.18). Проверила результат вручную, он сошелся с результатом программы.



```
kamilla@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
kamilla@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
kamilla@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 3
Введите значение переменной a: 4
Результат: 12
kamilla@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 2
Введите значение переменной a: 9
Результат: 18
kamilla@fedora:~/work/arch-pc/lab07$
```

Рис. 4.18: Запуск исполняемого файла



**Листинг. Программа для нахождения наименьшей из 3 целочисленных переменных A, B и C.**

```
““%include 'in_out.asm' section .data msg1 db 'Введите B:',0h msg2 db "Наимень-  
шее число:",0h A dd '26' C dd '68' 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' jl  
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' jl 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 'Наибольшее  
число:' mov eax,[min] call iprintLF ; Вывод 'max(A,B,C)' call quit ; Выход
```

\*Листинг. Программа, которая для введенных с клавиатуры значений A и B вычисляет значение

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

```
SECTION .data
```

```
msg1: DB 'Введите значение переменной x: ', 0
```

```
msg2: DB 'Введите значение переменной a: ', 0
```

```
msg3: DB 'Результат: ', 0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
a: RESB 80
```

```

SECTION .text
GLOBAL _start
_start:

mov eax, msg1
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov edi, eax

mov eax, msg2
call sprint
mov ecx, a
mov edx, 80
call sread
mov eax, a
call atoi
mov esi, eax

cmp esi, 8
jnl add_values
mov eax, esi
mov ebx, edi
mul ebx
jmp print_result

```

```
add_values:  
mov eax, esi  
add eax, 8
```

```
print_result:  
mov edi, eax  
mov eax, msg3  
call sprint  
mov eax, edi  
call iprintLF  
call quit
```

## **5 Выводы**

При выполнении данной лабораторной работы я изучила команды условного и безусловного перехода, приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файла листинга.

## **6 Список литературы**

1. Лабораторная работа №7