

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ  
НАРОДОВ**

**Факультет физико-математических и естественных наук  
Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7**

дисциплина: Архитектура компьютера

Студент: Силантьева Анастасия

Группа: НКАбд-05-25

№ ст. билета: 1032253541

**МОСКВА**

2025 г.

# Содержание

Список иллюстраций	3
1. Цель работы	4
2. Выполнение лабораторной работы	5
2.1 Реализация переходов в NASM	5
2.2 Изучение структуры файла листинга	9
2.3 Задания для самостоятельной работы	11
3. Вывод	15
Список литературы	16

## Список иллюстраций

1. Рис. 1 Создание каталога и файла для программы
2. Рис. 2 Код из листинга и сохранение программы
3. Рис. 3 Запущенный файл
4. Рис. 4 Измененный код
5. Рис. 5 Запуск измененной программы
6. Рис. 6 Измененная программа
7. Рис. 7 Проверка изменений
8. Рис. 8 Сохранение новой программы
9. Рис. 9 Проверка программы из листинга
10. Рис. 10 Проверка файла листинга
11. Рис. 11 Удаление операнда из кода
12. Рис. 11 Просмотр ошибки в файле листинга
13. Рис. 13 Первая программа самост. работы
14. Рис. 14 Проверка работы первой программы
15. Рис. 15 Вторая программа самостоятельной работы
16. Рис. 16 Проверка работы второй программы

## 1. Цель работы

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

## 2. Выполнение лабораторной работы

### 2.1 Реализация переходов в NASM

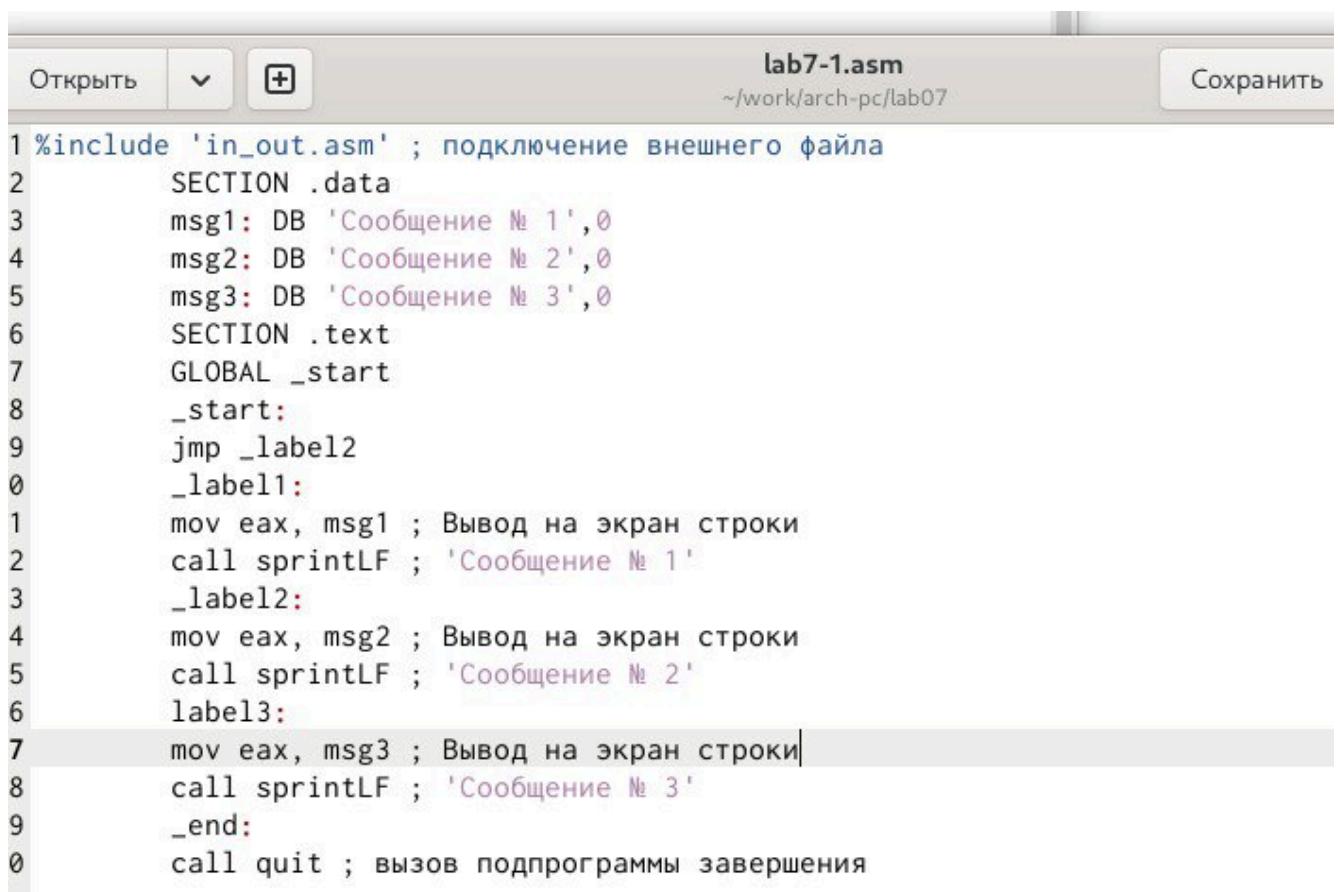
Создаю каталог для программ лабораторной работы №7, перехожу в него и создаю файл lab7-1.asm. [\(Рис. 1\)](#)



```
azsilantjeva@dk3n55 ~ $ mkdir ~/work/arch-pc/lab07
azsilantjeva@dk3n55 ~ $ cd ~/work/arch-pc/lab07
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ touch lab7-1.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 1 Создание каталога и файла для программы

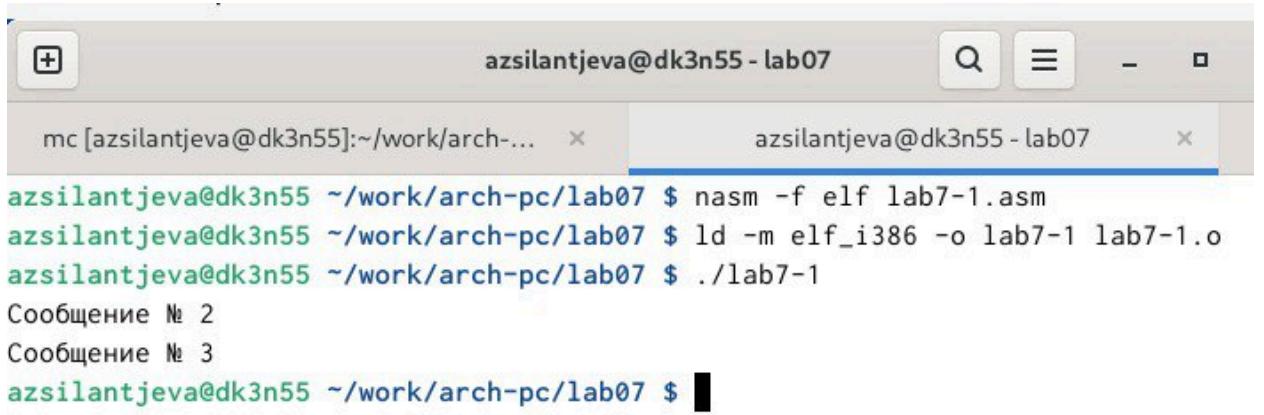
Копирую код из листинга в файл будущей программы.([Рис. 2](#))



```
Открыть ▾ + lab7-1.asm
~/work/arch-pc/lab07 Сохранить
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
0 _label1:
1 mov eax, msg1 ; Вывод на экран строки
2 call sprintLF ; 'Сообщение № 1'
3 _label2:
4 mov eax, msg2 ; Вывод на экран строки
5 call sprintLF ; 'Сообщение № 2'
6 _label3:
7 mov eax, msg3 ; Вывод на экран строки
8 call sprintLF ; 'Сообщение № 3'
9 _end:
0 call quit ; вызов подпрограммы завершения
```

Рис. 2 Код из листинга и сохранение программы

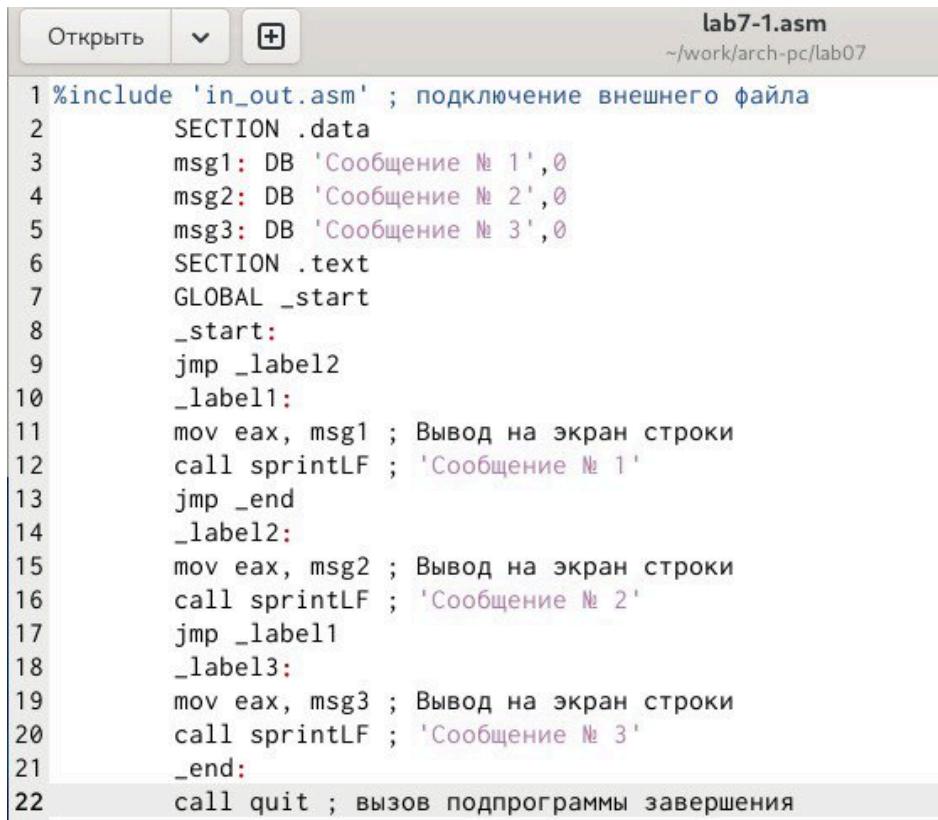
При запуске программы я убедилась в том, что неусловный переход действительно изменяет порядок выполнения инструкций ([рис. 3](#)).



```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 3
Сообщение № 1
```

Рис. 3 Запущенный файл

Изменяю программу таким образом, чтобы поменялся порядок выполнения функций ([рис. 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 sprintLF ; 'Сообщение № 1'
13 jmp _end
14 _label2:
15 mov eax, msg3 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 3'
17 jmp _label1
18 _label3:
19 mov eax, msg2 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 2'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4 Измененный код

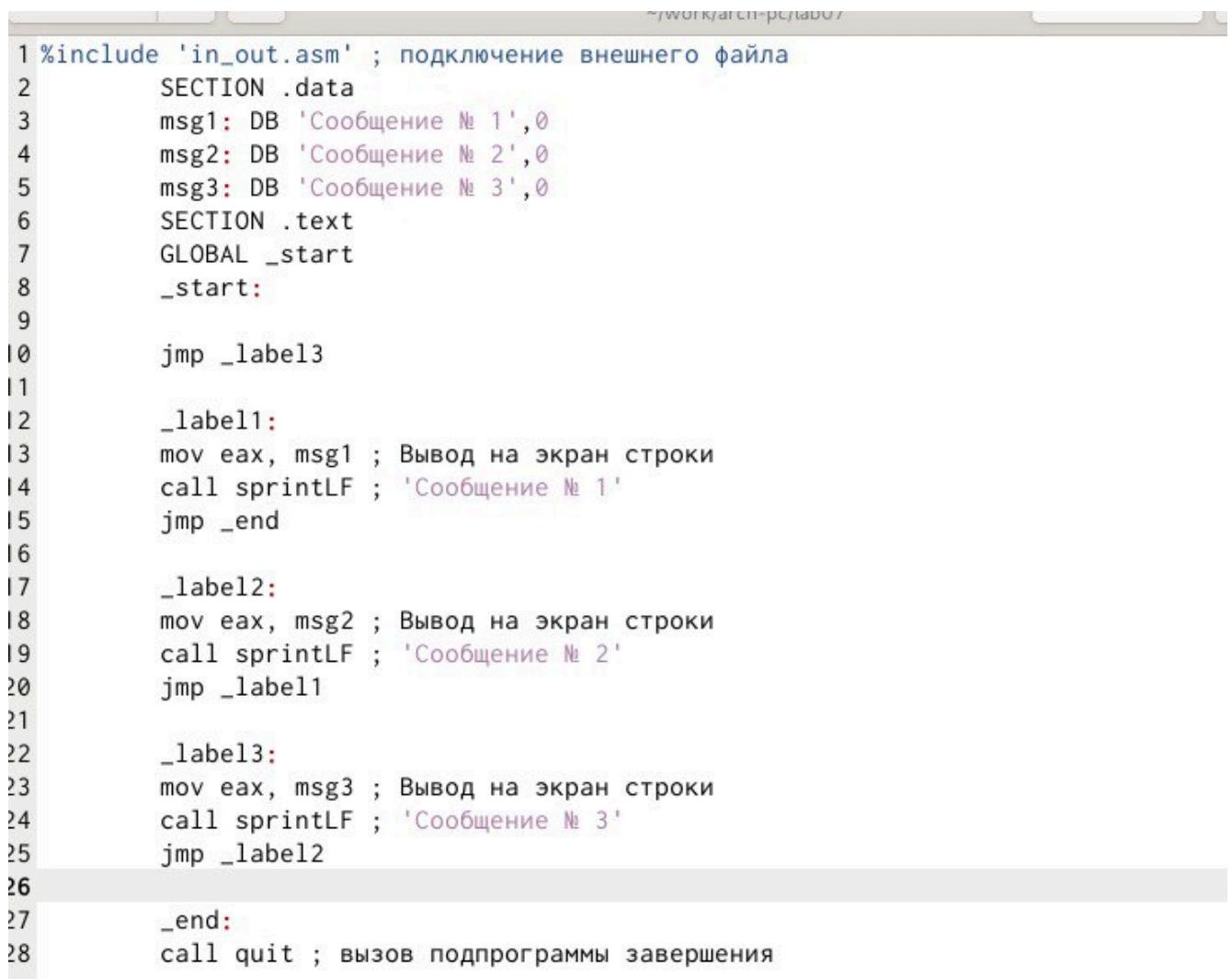
Запускаю программу и проверяю, что примененные изменения верны [\(рис. 5\)](#).

```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 2
Сообщение № 1
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 5 Запуск измененной программы

Теперь изменяю текст программы так, чтобы все три сообщения вывелись в обратном порядке [\(рис. 6\)](#).

```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```



The screenshot shows a terminal window with the following content:

```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

The assembly code is as follows:

```
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
10    jmp _label3
11
12    _label1:
13    mov eax, msg1 ; Вывод на экран строки
14    call sprintLF ; 'Сообщение № 1'
15    jmp _end
16
17    _label2:
18    mov eax, msg2 ; Вывод на экран строки
19    call sprintLF ; 'Сообщение № 2'
20    jmp _label1
21
22    _label3:
23    mov eax, msg3 ; Вывод на экран строки
24    call sprintLF ; 'Сообщение № 3'
25    jmp _label2
26
27    _end:
28    call quit ; вызов подпрограммы завершения
```

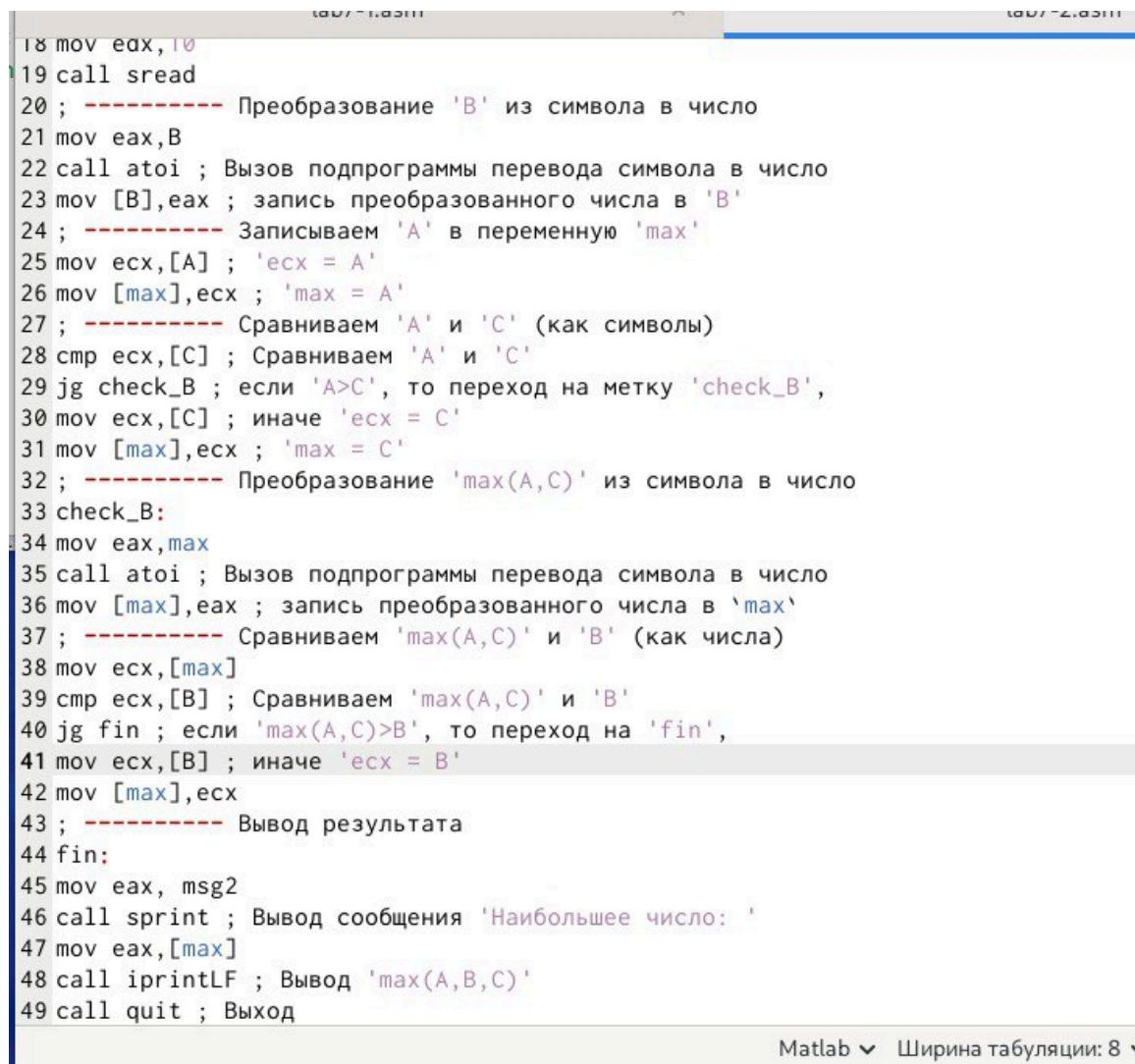
Рис. 6 Измененная программа

Работа выполнена корректно, программа в нужном мне порядке выводит сообщения (рис. 7).

```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 7 Проверка изменений

Создаю новый рабочий файл и вставляю в него код из следующего листинга (рис. 8).



```
18 mov eax,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 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
```

Matlab ▾ Ширина табуляции: 8 ▾

Рис. 8 Сохранение новой программы

Программа выводит значение переменной с максимальным значением, проверяю работу программы с разными входными данными (рис. 9).

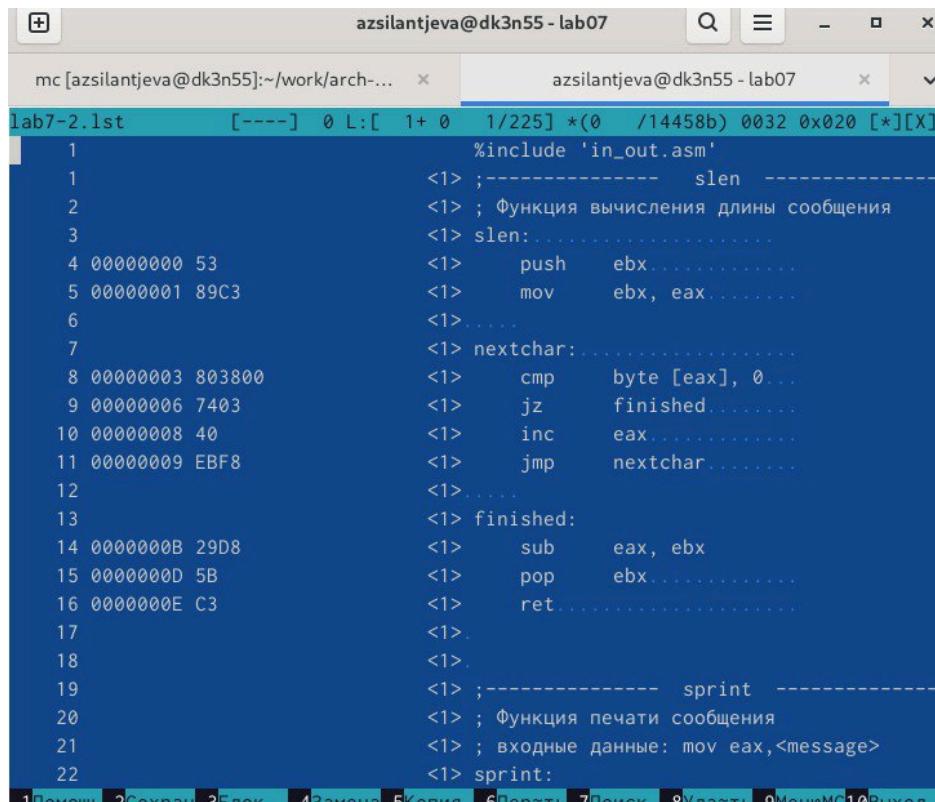


```
azsilantjeva@dk3n55 ~/work/arch-... x azsilantjeva@dk3n55 - lab07 x
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 25
Наибольшее число: 50
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 60
Наибольшее число: 60
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./lab7-2
Введите B: 10
Наибольшее число: 50
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 9 Проверка программы из листинга

## 2.2 Изучение структуры файла листинга

Создаю файл листинга с помощью ключа `-l`, команды nasm и открываю его с помощью текстового редактора mcedit (рис. 10).



```
azsilantjeva@dk3n55 - lab07 x
azsilantjeva@dk3n55 - lab07 x
lab7-2.lst      [---] 0 L:[ 1+ 0  1/225] *(0  /14458b) 0032 0x020 [*][X]
1
1
2
3
4 00000000 53
5 00000001 89C3
6
7
8 00000003 803800
9 00000006 7403
10 00000008 40
11 00000009 EBF8
12
13
14 0000000B 29D8
15 0000000D 5B
16 0000000E C3
17
18
19
20
21
22
```

include 'in\_out.asm'  
<1> ;----- slen -----  
<1> ; Функция вычисления длины сообщения  
<1> slen:  
<1> push ebx.....  
<1> mov ebx, eax.....  
<1>.....  
<1> nextchar:.....  
<1> cmp byte [eax], 0...  
<1> jz finished.....  
<1> inc eax.....  
<1> jmp nextchar.....  
<1>.....  
<1> finished:  
<1> sub eax, ebx  
<1> pop ebx.....  
<1> ret.....  
<1>.....  
<1>.....  
<1> ;----- sprint -----  
<1> ; Функция печати сообщения  
<1> ; входные данные: mov eax,<message>  
<1> sprint:

Рис. 10 Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может вовсе не совпадать с номером строки изначального файла. Второе вхождение - адрес, смещение машинного кода относительно начала текущего сегмента, затем непосредственно идет сам машинный код, а заключает строку исходный текст программы с комментариями.

Удаляю один operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем ([рис. 11](#)).

Рис. 11 Удаление операнда из кода

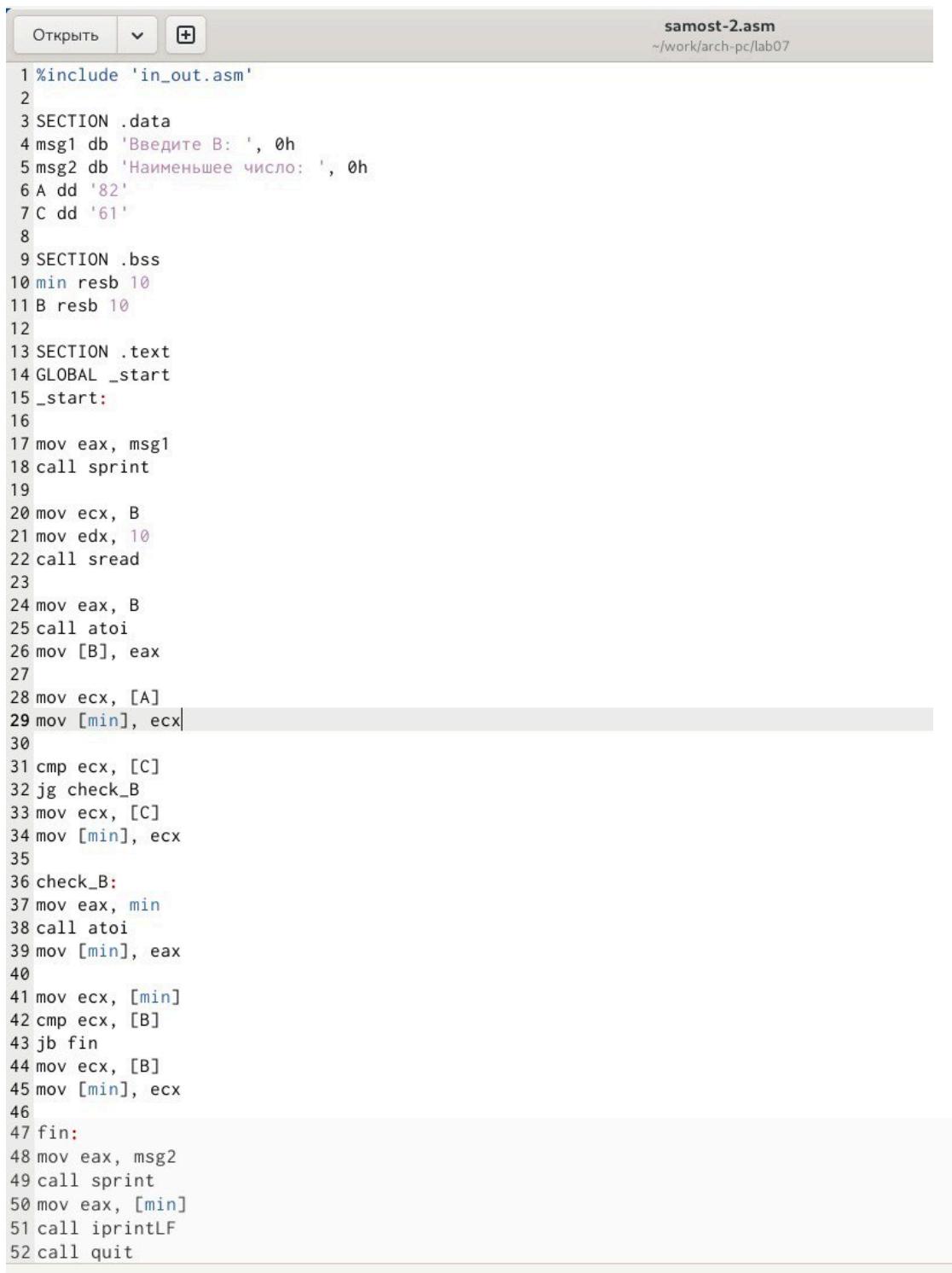
В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. ([рис. 12](#)).

```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf -l lab7-2.1st lab7-2.asm
lab7-2.asm:38: error: invalid combination of opcode and operands
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 12 Просмотр ошибки в файле листинга

### 2.3 Задания для самостоятельной работы

Я буду использовать свой вариант - второй - из предыдущей лабораторной работы. Возвращаю операнд к функции в программе и изменяю ее так, чтобы она выводила переменную с наименьшим значением ([рис. 13](#))



Notepad window showing assembly code for 'samost-2.asm' in the file path ' ~/work/arch-pc/lab07'. The code is as follows:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg1 db 'Введите В: ', 0h
5 msg2 db 'Наименьшее число: ', 0h
6 A dd '82'
7 C dd '61'
8
9 SECTION .bss
10 min resb 10
11 B resb 10
12
13 SECTION .text
14 GLOBAL _start
15 _start:
16
17 mov eax, msg1
18 call sprint
19
20 mov ecx, B
21 mov edx, 10
22 call sread
23
24 mov eax, B
25 call atoi
26 mov [B], eax
27
28 mov ecx, [A]
29 mov [min], ecx
30
31 cmp ecx, [C]
32 jg check_B
33 mov ecx, [C]
34 mov [min], ecx
35
36 check_B:
37 mov eax, min
38 call atoi
39 mov [min], eax
40
41 mov ecx, [min]
42 cmp ecx, [B]
43 jb fin
44 mov ecx, [B]
45 mov [min], ecx
46
47 fin:
48 mov eax, msg2
49 call sprint
50 mov eax, [min]
51 call iprintLF
52 call quit
```

Рис. 13 Первая программа самост. работы

Проверяю корректность написания первой программы ([рис. 14](#)).

```

azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ touch samost-2.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf samost-2.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o samost-2 samost-2.o
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./samost-2
Введите B: 59
Наименьшее число: 59
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $

```

Рис. 14 Проверка работы первой программы

Далее я пишу программу, которая будет вычислять значение заданной функции согласно моему варианту для введенных с клавиатуры переменных a и x ([рис. 15](#)).

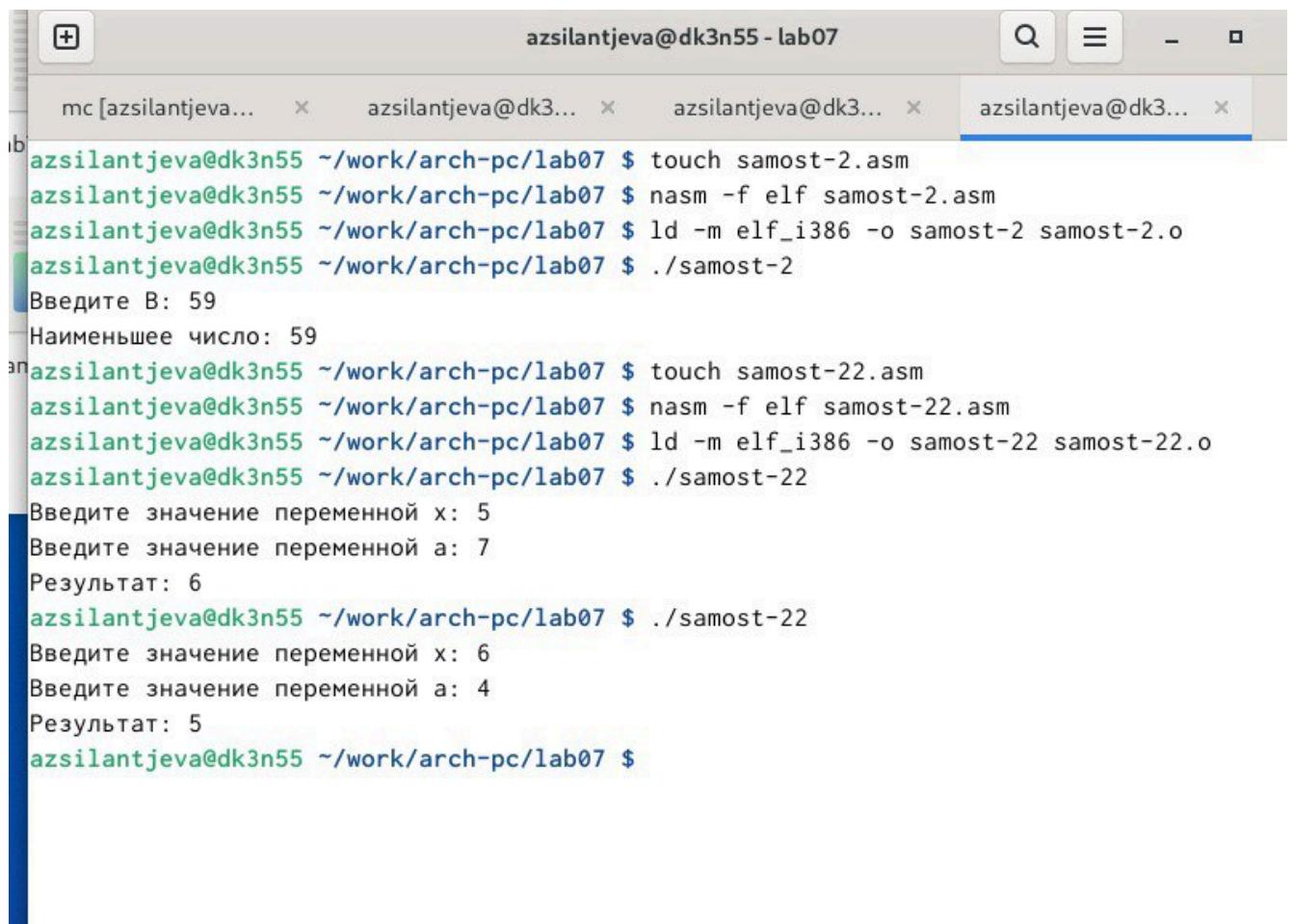
```

1 %include 'in_out.asm'
2
3 SECTION .data
4     msg_x: DB 'Введите значение переменной x: ', 0
5     msg_a: DB 'Введите значение переменной a: ', 0
6     res: DB 'Результат: ', 0
7
8 SECTION .bss
9     x: RESB 80
10    a: RESB 80
11
12 SECTION .text
13 GLOBAL _start
14
15 _start:
16     ; Ввод значения переменной x
17     mov eax, msg_x
18     call sprint
19     mov ecx, x
20     mov edx, 80
21     call sread
22     mov eax, x
23     call atoi
24     mov edi, eax      ; edi = x
25
26     ; Ввод значения переменной a
27     mov eax, msg_a
28     call sprint
29     mov ecx, a
30     mov edx, 80
31     call sread
32     mov eax, a
33     call atoi
34     mov esi, eax      ; esi = a
35
36     ; Сравниваем x и a
37     cmp edi, esi
38     jl less_than_a    ; если x < a, перейти к less_than_a
39
40     ; Если x >= a, вычисляем x - 1
41     dec edi          ; edi = x - 1
42     jmp print_result
43
44 less_than_a:
45     ; Если x < a, вычисляем a - 1
46     dec esi          ; esi = a - 1
47     mov edi, esi      ; сохраняем результат в edi

```

Рис. 15 Вторая программа самостоятельной работы

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений а и х ([рис. 16](#)).



The screenshot shows a terminal window with the title bar "azsilantjeva@dk3n55 - lab07". The terminal contains the following text:

```
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ touch samost-2.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf samost-2.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o samost-2 samost-2.o
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./samost-2
Введите В: 59
Наименьшее число: 59
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ touch samost-22.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ nasm -f elf samost-22.asm
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o samost-22 samost-22.o
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./samost-22
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 6
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $ ./samost-22
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 5
azsilantjeva@dk3n55 ~/work/arch-pc/lab07 $
```

Рис. 16 Проверка работы второй программы

### 3. Вывод

При выполнении лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с использованием переходов, познакомилась с назначением и структурой файлов листинга.

## Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. —
6. 354 c. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
7. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 c. — ISBN 978-1491941591.
8. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
9. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 c. — ISBN 9781784396879.
10. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
11. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
12. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
13. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
14. Робачевский А., Немлюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВ Петербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
15. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: [http://www.stolyarov.info/books/asm\\_unix](http://www.stolyarov.info/books/asm_unix).
16. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
17. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science).