

# **Отчет по лабораторной работе №7**

**Группа: НКАбд-04-23**  
**Зоригоо Номун**

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
4.1	Реализация переходов в NASM . . . . .	7
4.2	Изучение структуры файлы листинга.....	10
4.3	Задания для самостоятельной работы.....	12
<b>5</b>	<b>Выводы</b>	<b>15</b>
	<b>Список литературы</b>	<b>16</b>

## Список иллюстраций

4.1	Создание файлов для лабораторной работы . . . . .	7
4.2	Ввод текста программы из листинга 7.1 . . . . .	7
4.3	Запуск программного кода . . . . .	8
4.4	Изменение текста программы . . . . .	8
4.5	Создание исполняемого файла . . . . .	8
4.6	Изменение текста программы . . . . .	9
4.7	Вывод программы . . . . .	9
4.8	Создание файла . . . . .	9
4.9	Ввод текста программы из листинга 7.3 .....	10
4.10	Проверка работы файла .....	10
4.11	Создание файла листинга .....	10
4.12	Изучение файла листинга .....	11
4.13	Выбранные строки файла .....	11
4.14	Удаление выделенного операнда из кода .....	12
4.15	Получение файла листинга .....	12
4.16	Написание программы .....	13
4.17	Запуск файла и проверка его работы .....	13
4.18	Написание программы .....	14
4.19	Запуск файла и проверка его работы .....	14

# **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

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.

Безусловный переход выполняется инструкцией `jmp`. Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

Листинг (в рамках понятийного аппарата NASM) — это один из выходных файлов, создаваемых транслятором. Он имеет текстовый вид и нужен при отладке программы, так как кроме строк самой программы он содержит дополнительную информацию.

## 4 Выполнение лабораторной работы

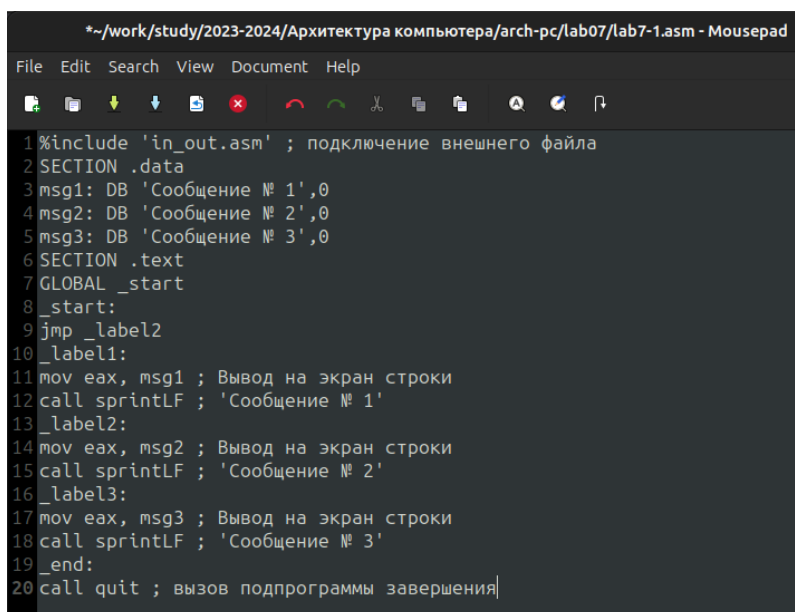
### 4.1 Реализация переходов в NASM

Я создаю каталог для программы лабораторной работы № 7 и перехожу в него и создаю файл lab7-1.asm. (рис. 4.1).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ mkdir lab07
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc$ cd lab07
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
```

Рис. 4.1: Создание файлов для лабораторной работы

Я скопировала lab7-1.asm текст программы из листинга 7.1 в Mousepad (рис. 4.2).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
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: Ввод текста программы из листинга 7.1

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

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-
pc/lab07$ nasm -f elf lab7-1.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-
pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-
pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
```

Рис. 4.3: Запуск программного кода

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого изменяю текст программы в соответствии с листингом 7.2. (рис. 4.4).

```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
[Icons]
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, msg2 ; Вывод на экран строки
16 call sprintLF ; 'Сообщение № 2'
17 jmp _label1
18 _label3:
19 mov eax, msg3 ; Вывод на экран строки
20 call sprintLF ; 'Сообщение № 3'
21 _end:
22 call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение текста программы

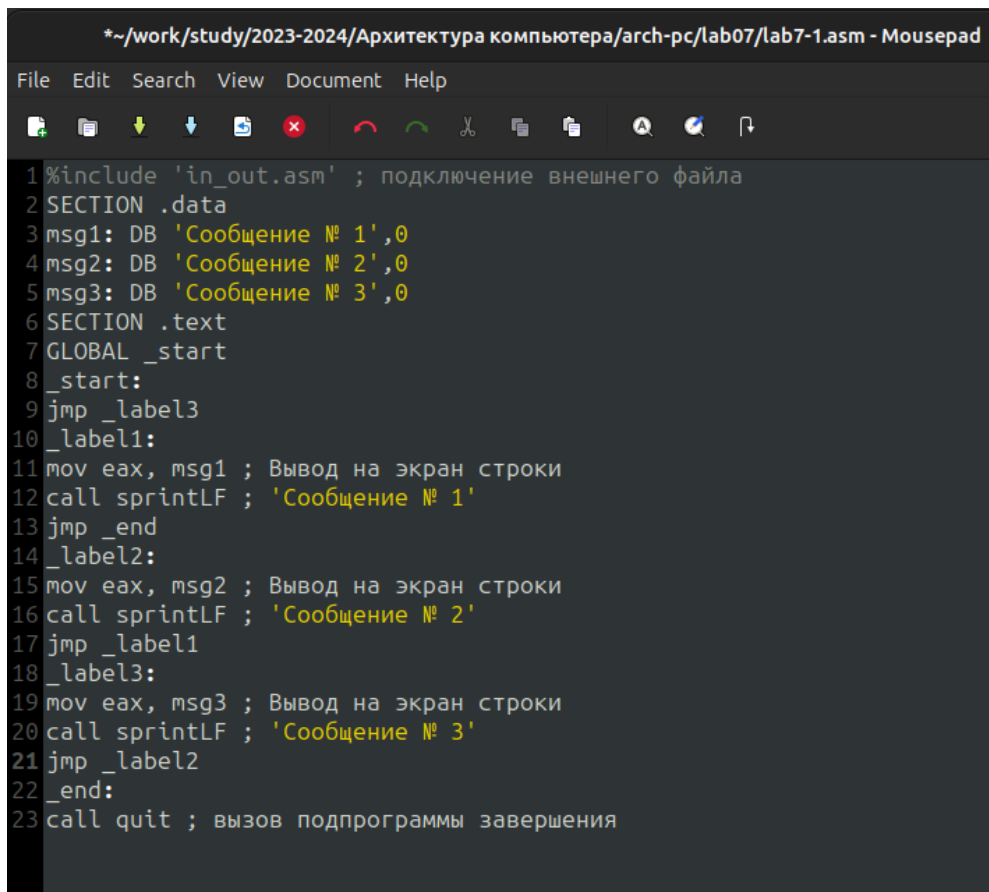


Создаю исполняемый файл и проверяю его работу. (рис. 4.5).

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

Рис. 4.5: Создание исполняемого файла

Затем изменяю текст программы, добавив в начале программы `jmp _label3`, `jmp _label2` в конце метки `jmp _label3`, `jmp _label1` добавляю в конце метки `jmp _label2`, и добавляю `jmp _end` в конце метки `jmp _label1`, (рис. 4.1).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
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: Изменение текста программы

чтобы вывод программы был следующим: (рис. 4.6).

```
ch-pc/lab07$ nasm -f elf lab7-1.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
```

Рис. 4.7: Вывод программы

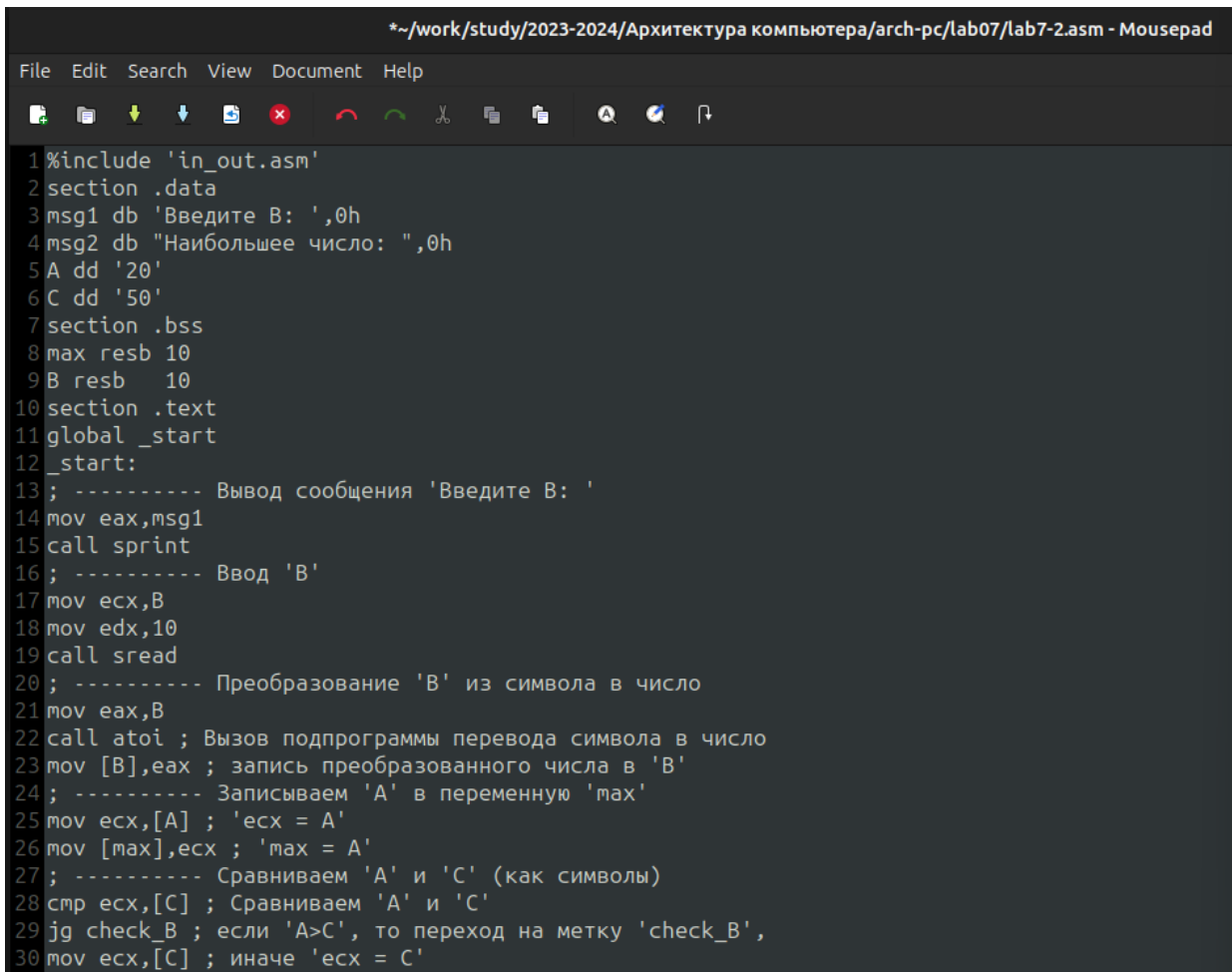
Рассмотрим программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: А,В и С. Значения для А и С задаются в программе, значение В вводится с клавиатуры.

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. 4.8).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ touch lab7-2.asm
```

Рис. 4.8: Создание файла

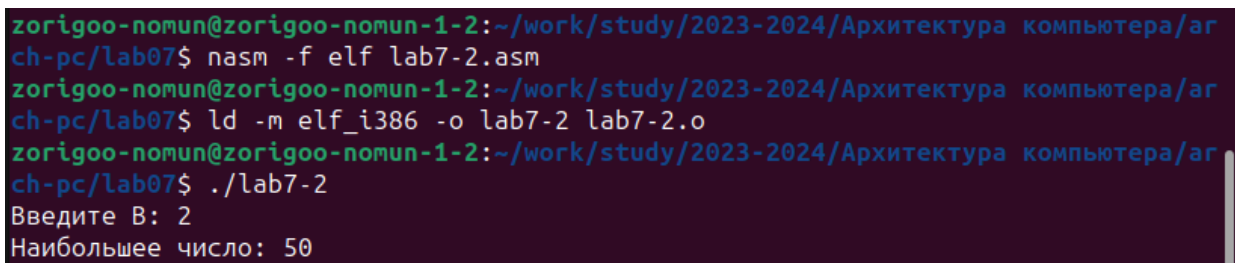
Текст программы из листинга 7.3 ввожу в lab7-2.asm. (рис. 4.9).



```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/lab7-2.asm - Mousepad
File Edit Search View Document Help
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 ; ----- 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'
```

Рис. 4.9: Ввод текста программы из листинга 7.3

Создаю исполняемый файл и проверяю его работу. (рис. 4.10).



```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 2
Наибольшее число: 50
```

Рис. 4.10: Проверка работы файла

Файл работает корректно.

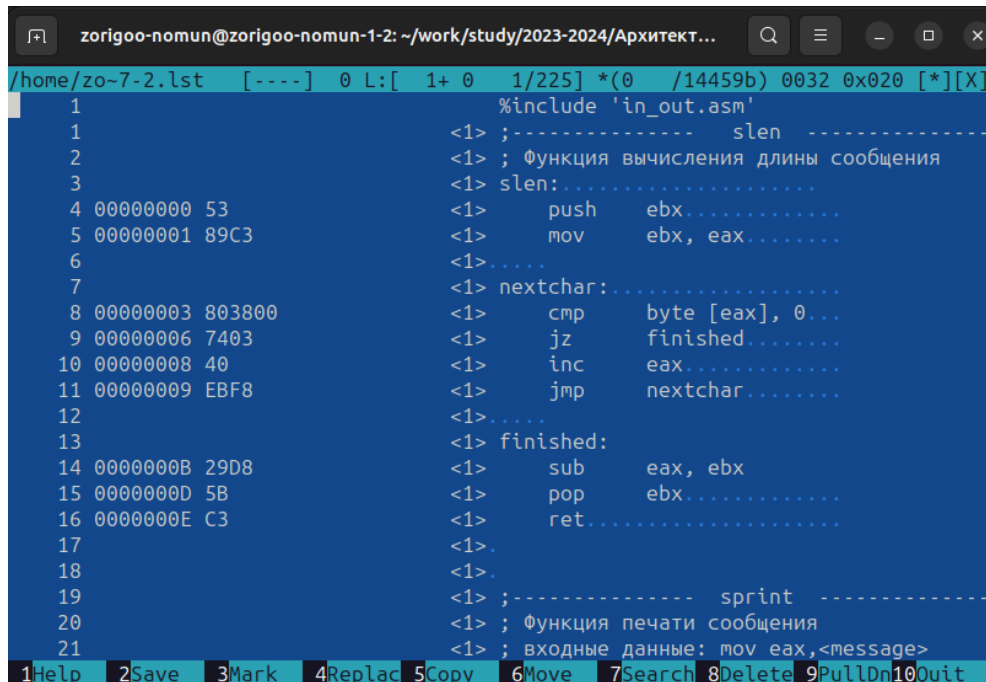
## 4.2 Изучение структуры файлы листинга

Создаю файл листинга для программы из файла lab7-2.asm. (рис. 4.11).

```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
```

Рис. 4.11: Создание файла листинга

Открываю файл листинга lab7-2.lst с помощью текстового редактора и внимательно изучаю его формат и содержимое. (рис. 4.12).



```
/home/zo-7-2.lst [----] 0 L: [ 1+ 0 1/225] *(0 /14459b) 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>
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 4.12: Изучение файла листинга

В представленных трех строчках содержатся следующие данные: (рис. 4.13).



```
2                               <1> ;----- slen -----
3                               <1> ; Функция вычисления длины сообщения
4                               <1> slen:.....
```

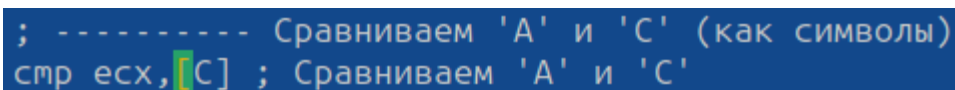
Рис. 4.13: Выбранные строки файла

“2” - номер строки кода, “; Функция вычисления длинны сообщения” - комментарий к коду, не имеет адреса и машинного кода.

“3” - номер строки кода, “slen” - название функции, не имеет адреса и машинного кода.

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

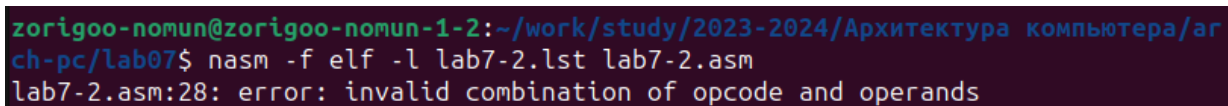
Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис. 4.14).



```
; ----- Сравниваем 'A' и 'C' (как символы)
cmp есх, [C] ; Сравниваем 'A' и 'C'
```

Рис. 4.14: Удаление выделенного операнда из кода

Выполняю трансляцию с получением файла листинга. (рис. 4.15).



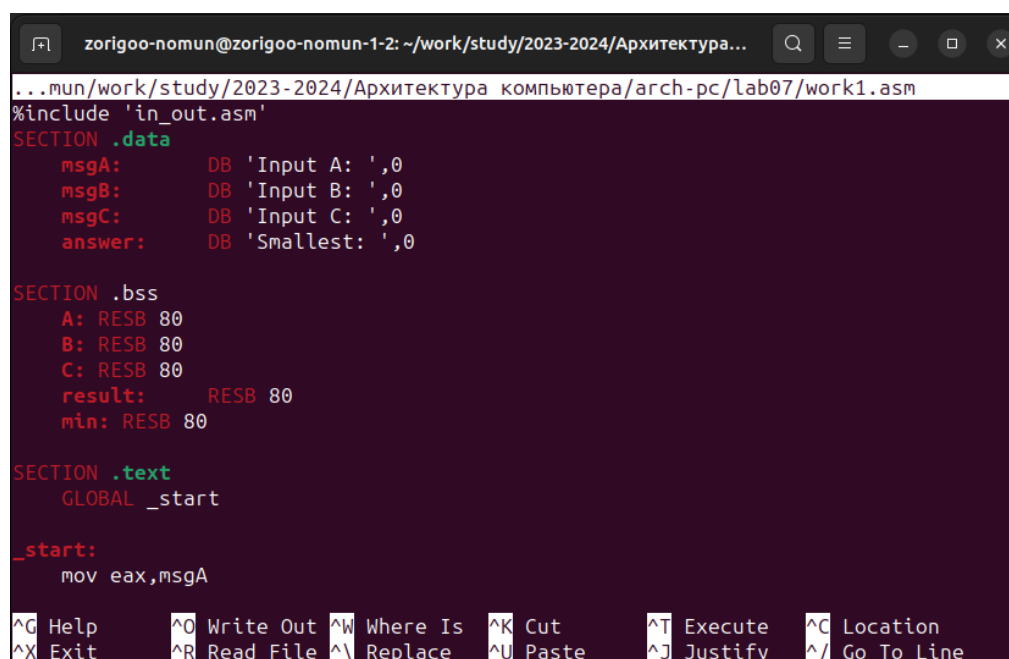
```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-
ch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:28: error: invalid combination of opcode and operands
```

Рис. 4.15: Получение файла листинга

На выходе я не получаю ни одного файла из-за ошибки: инструкция mov (единственная в коде содержит два операнда) не может работать, имея только один операнд, из-за чего нарушается работа кода.

## 4.3 Задания для самостоятельной работы

1. Пишу программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбираю из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант под номером 17, поэтому мои значения - 26, 12 и 68. (рис. 4.16).



```
zorigoo-nomun@zorigoo-nomun-1-2: ~/work/study/2023-2024/Архитектура...
...mun/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/work1.asm
%include 'in_out.asm'
SECTION .data
    msgA:      DB 'Input A: ',0
    msgB:      DB 'Input B: ',0
    msgC:      DB 'Input C: ',0
    answer:    DB 'Smallest: ',0

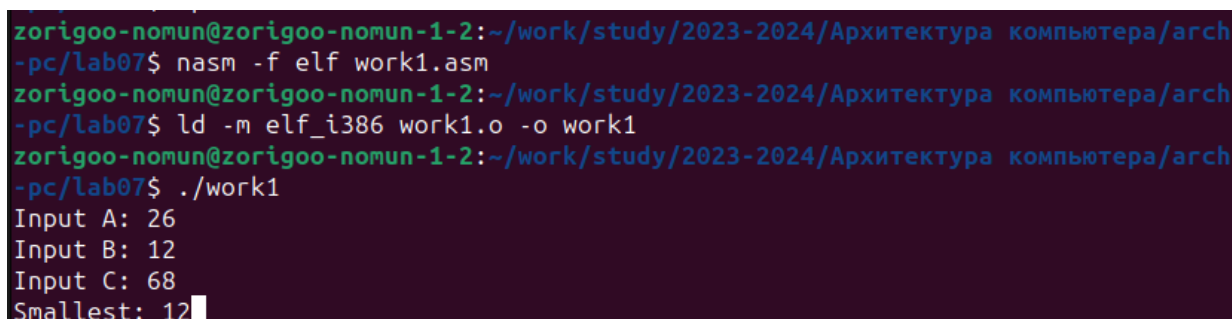
SECTION .bss
    A: RESB 80
    B: RESB 80
    C: RESB 80
    result:    RESB 80
    min: RESB 80

SECTION .text
    GLOBAL _start

_start:
    mov eax,msgA
```

Рис. 4.16: Написание программы

Создаю исполняемый файл и проверяю его работу, подставляя необходимые значение. (рис. 4.17).



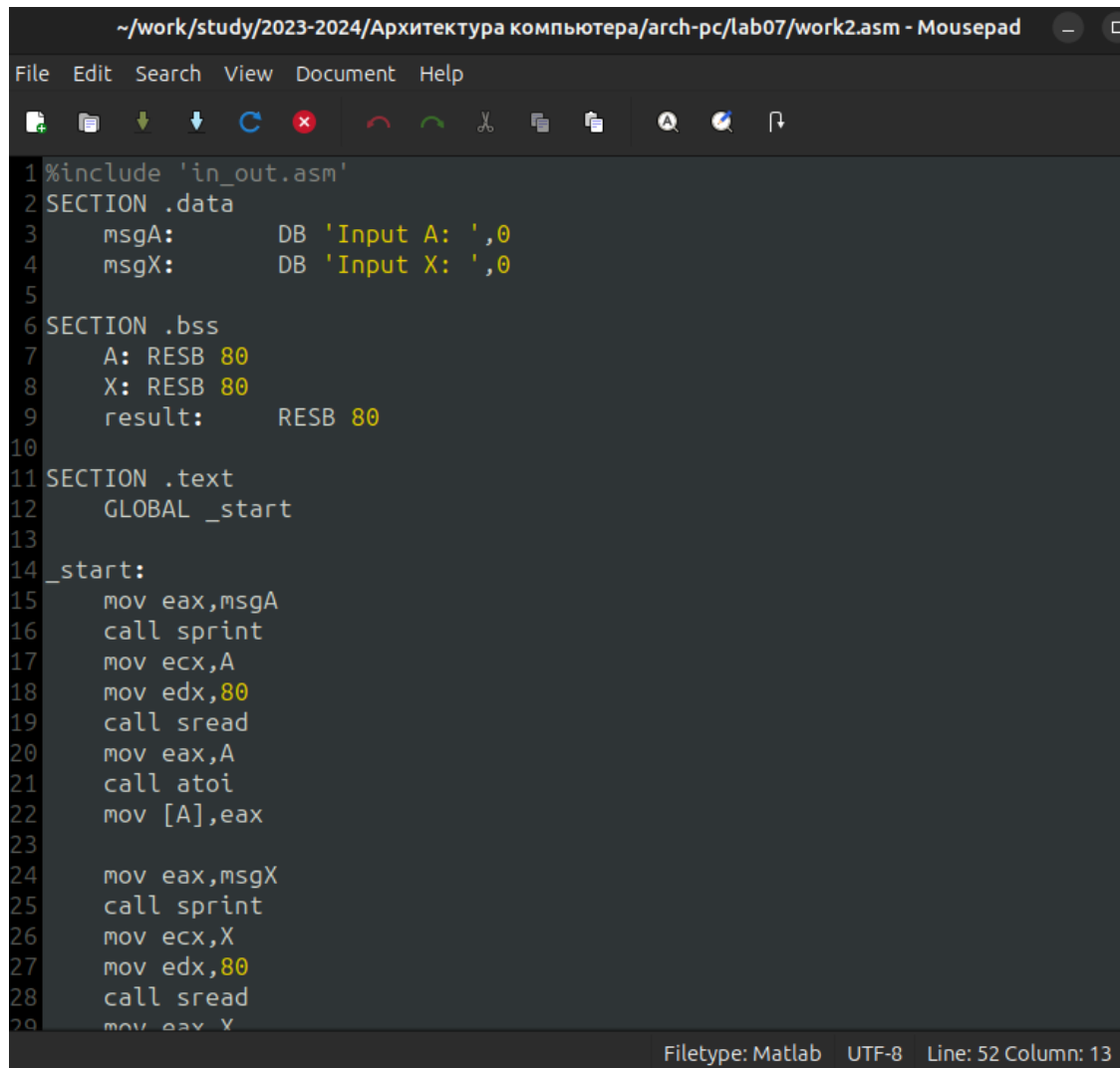
```
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ nasm -f elf work1.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 work1.o -o work1
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07$ ./work1
Input A: 26
Input B: 12
Input C: 68
Smallest: 12
```

Рис. 4.17: Запуск файла и проверка его работы

2. Пишу программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение и выводит результат вычислений заданной для моего варианта функции  $f(x)$ :

$$a + 8, a < 8$$

$$ax, \quad a \geq 8 \text{ (рис 4.18)}$$



```
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab07/work2.asm - Mousepad
File Edit Search View Document Help
1 %include 'in_out.asm'
2 SECTION .data
3     msgA:      DB 'Input A: ',0
4     msgX:      DB 'Input X: ',0
5
6 SECTION .bss
7     A: RESB 80
8     X: RESB 80
9     result:    RESB 80
10
11 SECTION .text
12     GLOBAL _start
13
14 _start:
15     mov eax,msgA
16     call sprint
17     mov ecx,A
18     mov edx,80
19     call sread
20     mov eax,A
21     call atoi
22     mov [A],eax
23
24     mov eax,msgX
25     call sprint
26     mov ecx,X
27     mov edx,80
28     call sread
29     mov eax,X
```

Filetype: Matlab UTF-8 Line: 52 Column: 13

Рис. 4.18: Написание программы

Создаю исполняемый файл и проверяю его работу для значений x и a соответственно: (3;4), (2;9). (рис. 4.19).

```
ch-pc/lab07$ nasm -f elf work2.asm
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/ар
ch-pc/lab07$ ld -m elf_i386 work2.o -o work2
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/ар
ch-pc/lab07$ ./work2
Input A: 3
Input X: 4
16
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/ар
ch-pc/lab07$ ./work2
Input A: 2
Input X: 9
17
zorigoo-nomun@zorigoo-nomun-1-2:~/work/study/2023-2024/Архитектура компьютера/арch-pc/lab07
$
```

Рис. 4.19: Запуск файла и проверка его работы

Программа работает корректно.



## 5 Выводы

Здесь кратко описываются итоги проделанной работы. По итогам данной лабораторной работы я изучила команды условного и безусловного переходов, приобрела навыки написания программ с использованием переходов и ознакомилась с назначением и структурой файла листинга, что поможет мне при выполнении последующих лабораторных работ.

# Список литературы

## 1. Архитектура ЭВМ