

Отчет по лабораторной работе №7

Группа: НКАбд-04-23

Монхжаргал

Тувшинбаяр

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	10
4.3	Задания для самостоятельной работы	12
5	Выводы	15
	Список литературы	16

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.

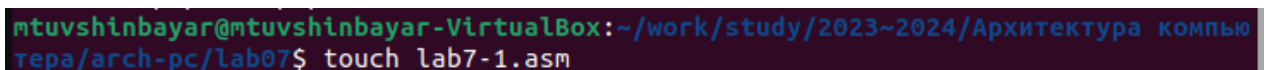
Безусловный переход выполняется инструкцией `jmp`. Инструкция `cmp` является одной из инструкций, которая позволяет сравнить операнды и выставляет флаги в зависимости от результата сравнения. Инструкция `cmp` является командой сравнения двух операндов и имеет такой же формат, как и команда вычитания.

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

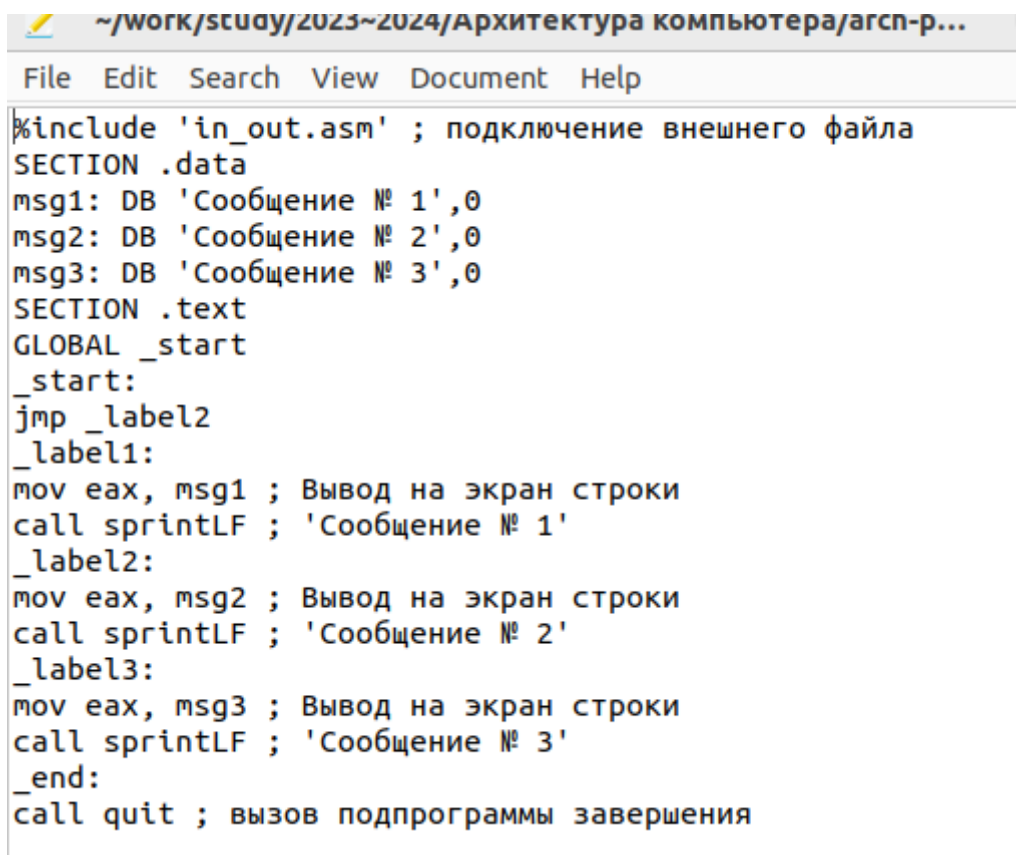
Я создаю каталог для программы лабораторной работы № 7 и перехожу в него и создаю файл lab7-1.asm. (рис. 4.1).



```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компью  
тера/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 Mouseрад (рис.4.2).

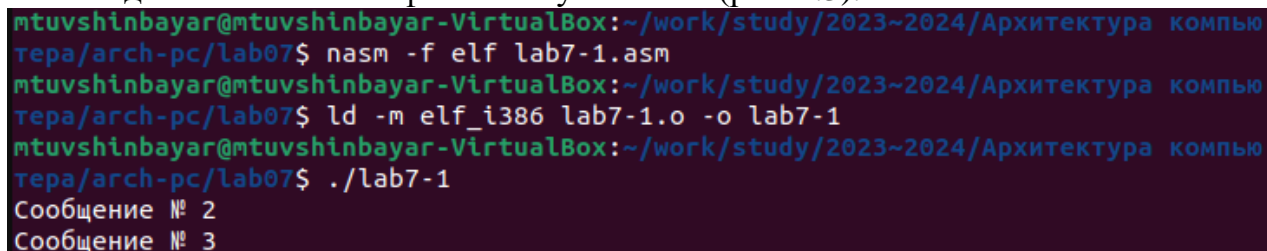


The screenshot shows a text editor window with a menu bar (File, Edit, Search, View, Document, Help) and a title bar indicating the file path: ~/work/study/2023~2024/Архитектура компьютера/арсп-р... The editor contains the following assembly code:

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.2: Ввод текста программы из листинга 7.1

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

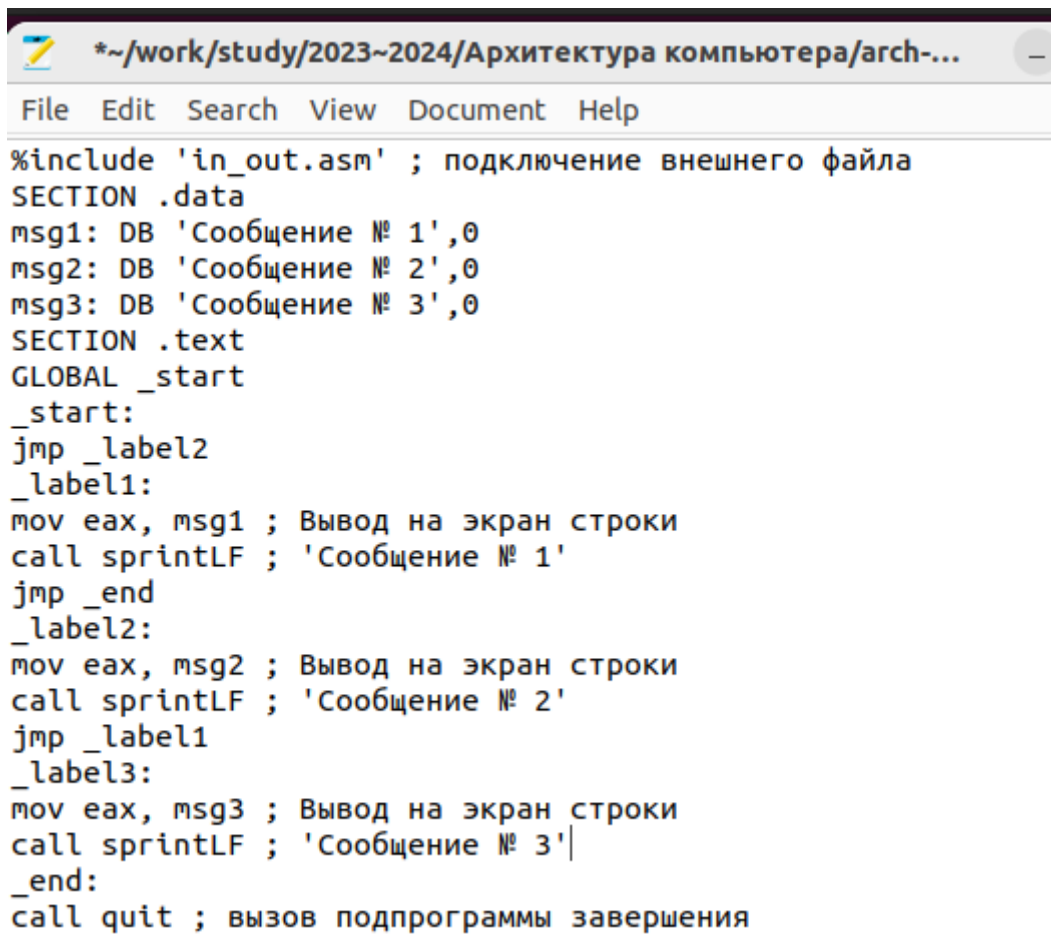


```
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023~2024/Архитектура компь  
тера/arch-pc/lab07$ nasm -f elf lab7-1.asm  
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023~2024/Архитектура компь  
тера/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1  
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023~2024/Архитектура компь  
тера/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).



The image shows a screenshot of a text editor window. The title bar at the top reads `*~/work/study/2023~2024/Архитектура компьютера/arch-...`. Below the title bar is a menu bar with the following items: File, Edit, Search, View, Document, and Help. The main text area contains assembly code in Russian. The code defines three data messages, sets up a text section, and contains a loop that prints each message to the screen. The code is as follows:

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 4.4: Изменение текста программы

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

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

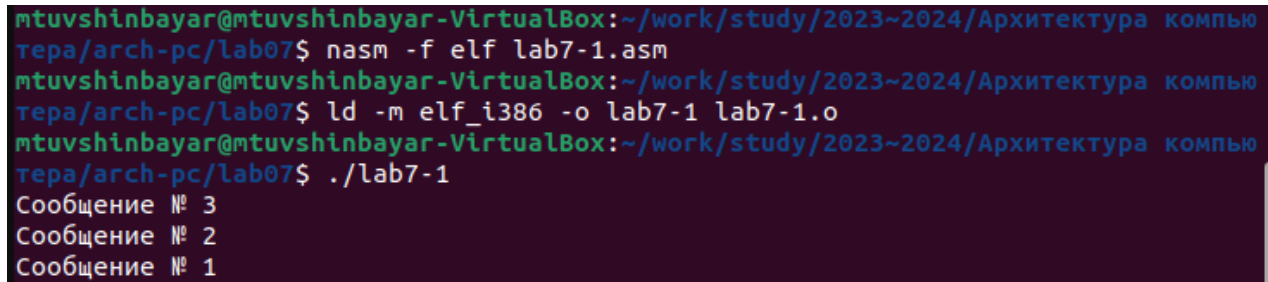
Рис. 4.5: Создание исполняемого файла

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).

```
File Edit Search View Document Help  
%include 'in_out.asm' ; подключение внешнего файла  
SECTION .data  
msg1: DB 'Сообщение № 1',0  
msg2: DB 'Сообщение № 2',0  
msg3: DB 'Сообщение № 3',0  
SECTION .text  
GLOBAL _start  
_start:  
jmp _label3  
_label1:  
mov eax, msg1 ; Вывод на экран строки  
call sprintf ; 'Сообщение № 1'  
jmp _end  
_label2:  
mov eax, msg2 ; Вывод на экран строки  
call sprintf ; 'Сообщение № 2'  
jmp _label1  
_label3:  
mov eax, msg3 ; Вывод на экран строки  
call sprintf ; 'Сообщение № 3'  
jmp _label2  
_end:  
call quit ; вызов подпрограммы завершения
```

Рис. 4.6: Изменение текста программы

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

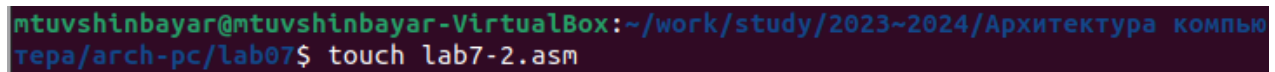


```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компьтера/arch-pc/lab07$ nasm -f elf lab7-1.asm
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компьтера/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компьтера/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.

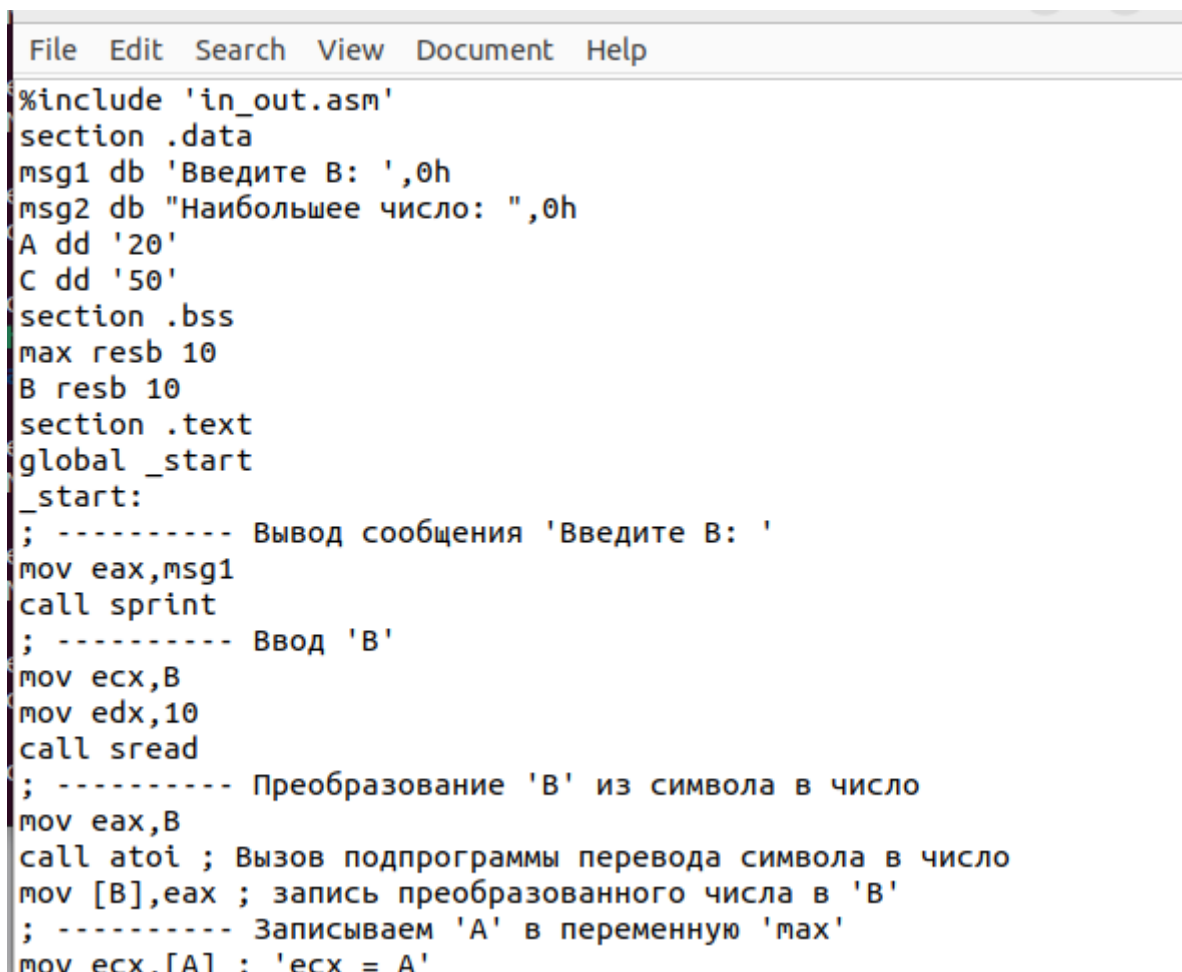
Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. (рис. 4.8).



```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компьтера/arch-pc/lab07$ touch lab7-2.asm
```

Рис. 4.8: Создание файла

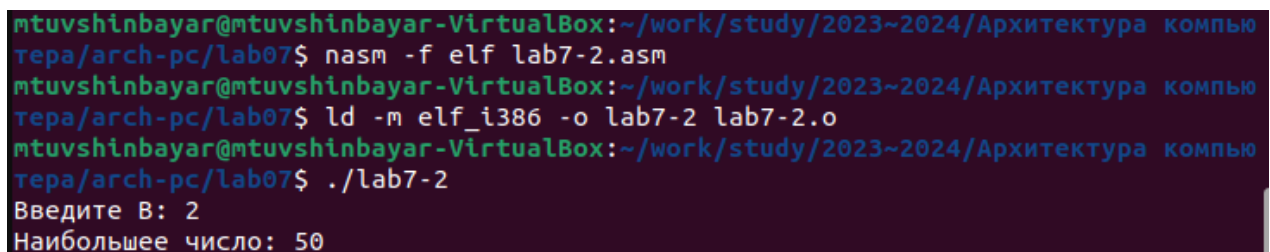
Текст программы из листинга 7.3 ввожу в lab7-2.asm. (рис. 4.9).



```
File Edit Search View Document Help
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наибольшее число: ",0h
A dd '20'
C dd '50'
section .bss
max 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 'max'
mov ecx,[A] ; 'ecx = A'
```

Рис. 4.9: Ввод текста программы из листинга 7.3

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



```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компью
тера/arch-pc/lab07$ nasm -f elf lab7-2.asm
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компью
тера/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компью
тера/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).

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023~2024/Архитектура компь  
тера/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 внима-
тельно изучаю его формат и содержимое. (рис. 4.12).

Рис. 4.12: Изучение файла листинга

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

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

Рис. 4.13: Выбранные строки файла

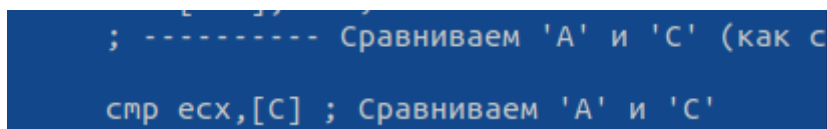
“2” - номер строки кода, “; Функция вычисления длинны сообщения” - коммен-
тарий к коду, не имеет адреса и машинного кода.

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

“4” - номер строки кода, “00000000” - адрес строки, “53” - машинный код, “push

ebx” - исходный текст программы, инструкция “push” помещает операнд “ebx” в стек.

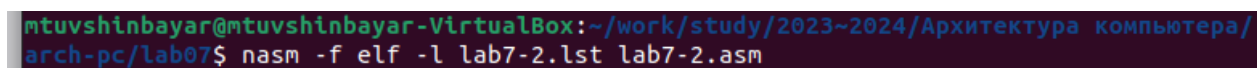
Открываю файл с программой lab7-2.asm и в выбранной мной инструкции с двумя операндами удаляю выделенный операнд. (рис. 4.14).



```
; ----- Сравниваем 'А' и 'С' (как с  
cmp ecx, [C] ; Сравниваем 'А' и 'С'
```

Рис. 4.14: Удаление выделенного операнда из кода

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



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

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).

```
...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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 4.16: Написание программы

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

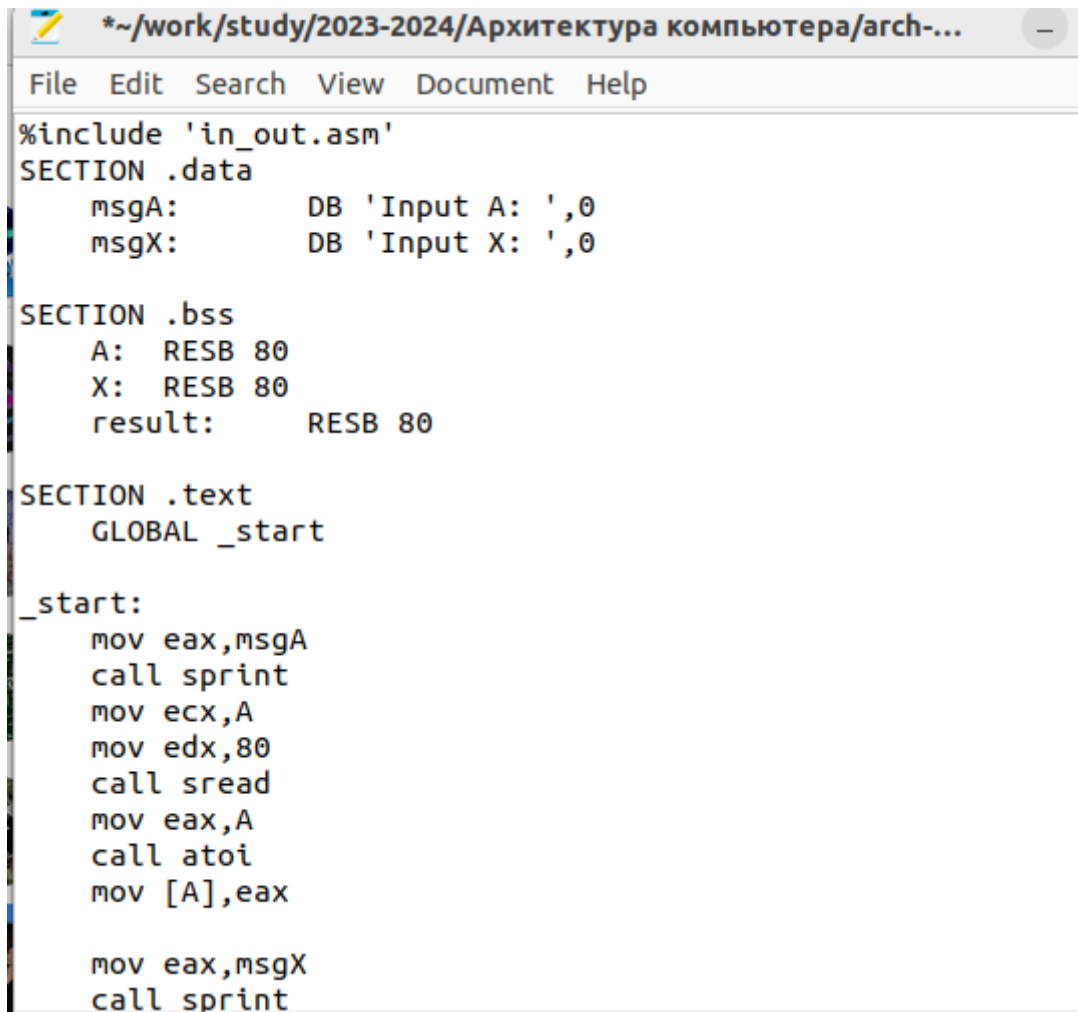
```
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компью
тера/arch-pc/lab07$ nasm -f elf work1.asm
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компью
тера/arch-pc/lab07$ ld -m elf_i386 work1.o -o work1
mtuvshinbayar@mtuvshinbayar-VirtualBox:~/work/study/2023-2024/Архитектура компью
тера/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 $f(x)$:

$a + 8, a < 8$

$ax, a \geq 8$ (рис 4.18)



```
*~/work/study/2023-2024/Архитектура компьютера/arch-...
File Edit Search View Document Help
%include 'in_out.asm'
SECTION .data
    msgA:      DB 'Input A: ',0
    msgX:      DB 'Input X: ',0

SECTION .bss
    A:  RESB 80
    X:  RESB 80
    result:  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
    mov [A],eax

    mov eax,msgX
    call sprint
```

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).

```
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютерных систем/repas/arch-pc/lab07$ nasm -f elf work2.asm
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютерных систем/repas/arch-pc/lab07$ ld -m elf_i386 work2.o -o work2
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютерных систем/repas/arch-pc/lab07$ ./work2
Input A: 3
Input X: 4
16
mtuvshinbayer@mtuvshinbayer-VirtualBox:~/work/study/2023-2024/Архитектура компьютерных систем/repas/arch-pc/lab07$ ./work2
Input A: 2
Input X: 9
17
```

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