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: Юсупова Амина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

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 файлов листинга
3. Самостоятельное написание программ по материалам лабораторной 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 про- граммы без каких-либо условий.

4 Выполнение лабораторной работы

4.1 Реализация переходов в NASM

Создаю каталог для программ лабораторной работы № 7, перехожу в него и создаю файл lab7-1.asm (рис. 1).

```
aryusupova@aryusupova-VirtualBox:~$ mkdir ~/work/arch-pc/lab07
aryusupova@aryusupova-VirtualBox:~$ cd ~/work/arch-pc/lab07
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-1.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 1: Создание каталога и файла lab7-1.asm в нём

Перехожу в файл lab7-1.asm и ввожу текст программы из листинга 7.1 (рис. 2).

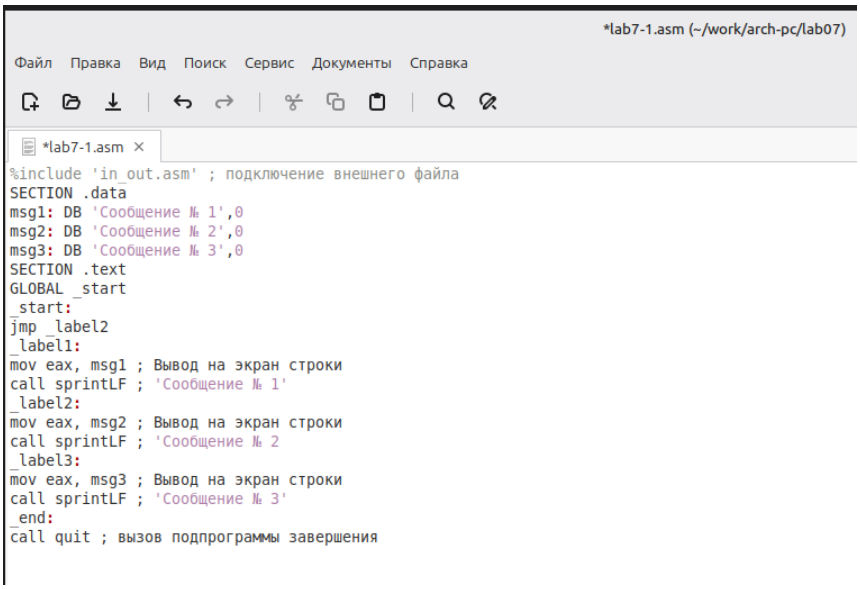

Рисунок 2: Ввод кода программы из листинга 7.1

Создала исполняемый файл и запустила его. Результат работы данной программы совпадает с указанным в задании. При запуске программы я убедилась в том, что неусловный переход действительно изменяет порядок выполнения инструкций (рис. 3).

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

Рисунок 3: Запуск программы

Изменяю программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавляю инструкцию jmp с меткой _label1 (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию jmp с меткой _end (т.е. переход к инструкции call quit). Изменяю текст программы в соответствии с листингом 7.2 (рис. 4).

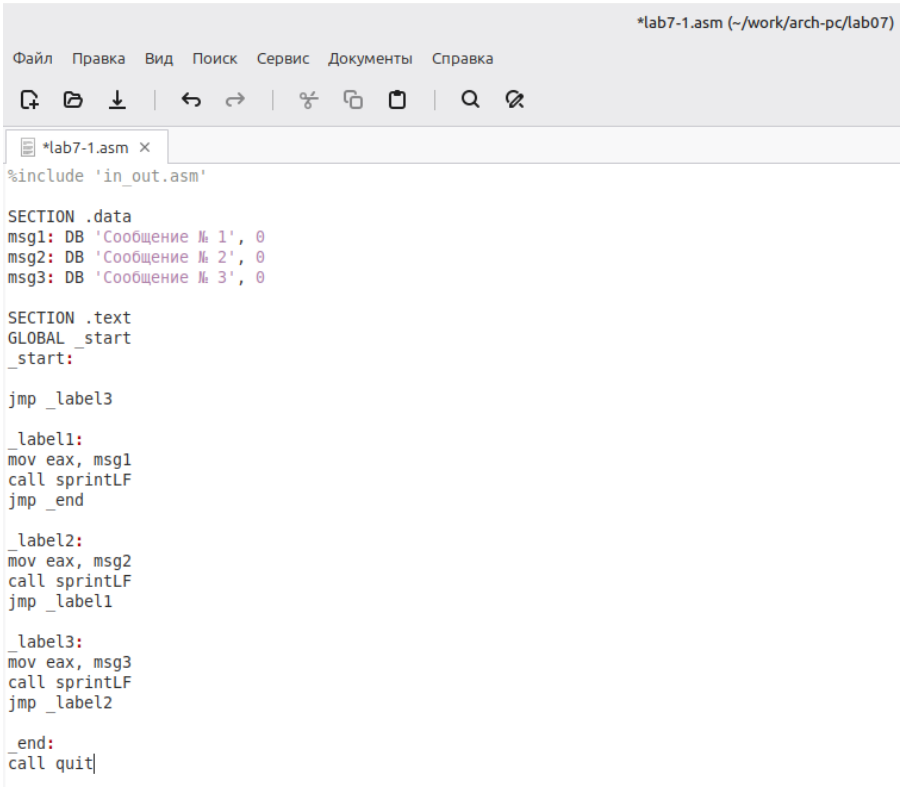

Рисунок 4: Изменение программы

Запускаю программу и проверяю, что примененные изменения верны. Вывод программы соответствует (рис. 5).

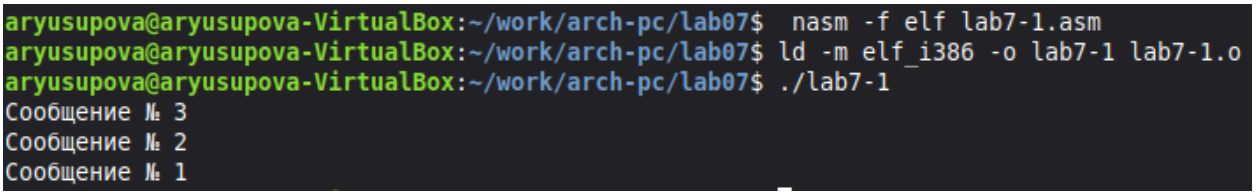

Рисунок 5: Запуск изменённой программы

Создаю файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 с помощью команды touch (рис. 6). И пишу код из листинга 7.3 (рис. 7). Программа выводит значение переменной с максимальным значением, проверяю работу программы, вводя разные значения (рис. 8).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-2.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 6: Создание файла lab7-2.asm

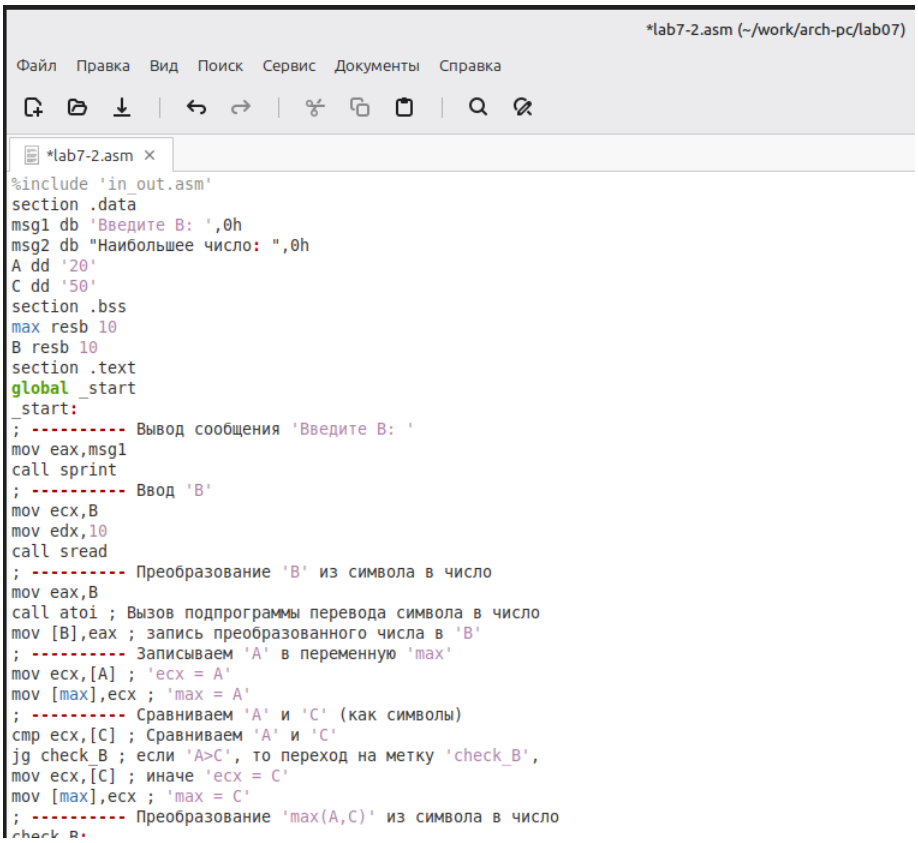

Рисунок 7: Ввод кода программы из листинга 7.3

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 23
Наибольшее число: 50
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 66
Наибольшее число: 66
```

Рисунок 8: Проверка программы на нахождения максимума из листинга

4.2 Изучение структуры файла листинга

Создаю файл листинга с помощью флага -l команды nasm и открываю его с помощью текстового редактора mcedit (рис. 9).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рисунок 9: Создание файла и открытие его

Проверяю файл листинга (рис.10).

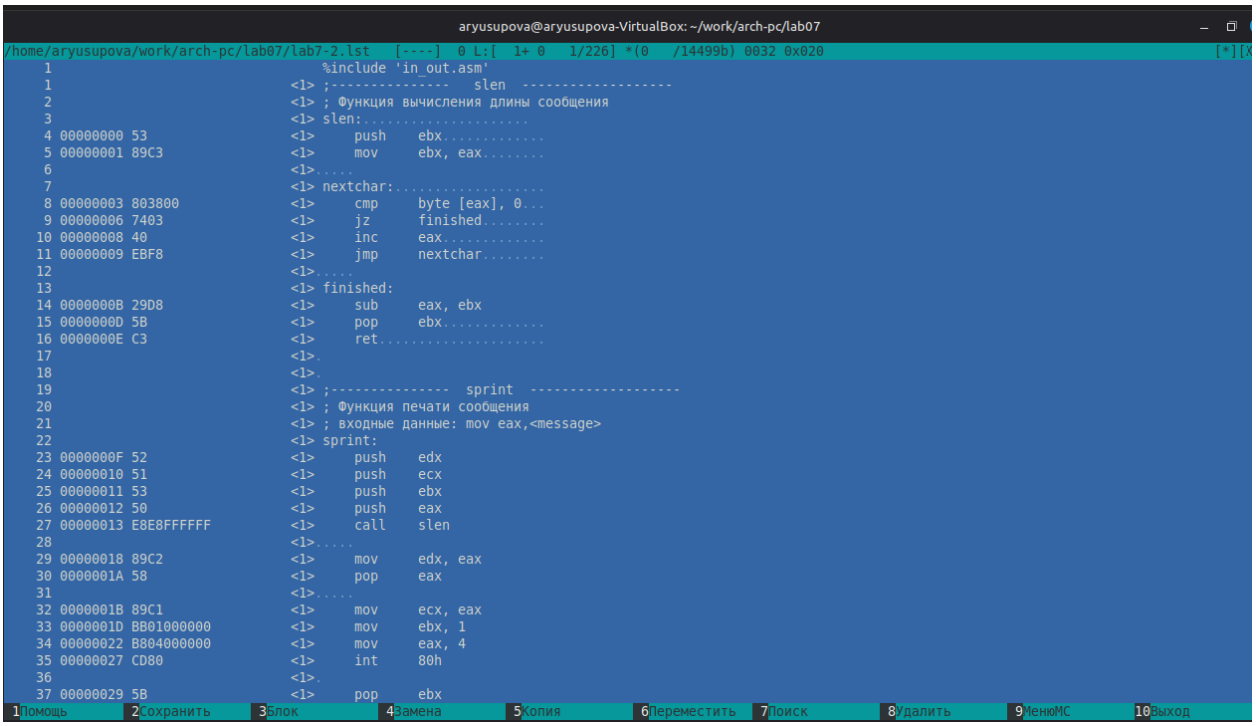

Рисунок 10: Проверка файла листинга

Первое значение в файле листинга - номер строки, и он может 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 операнд из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

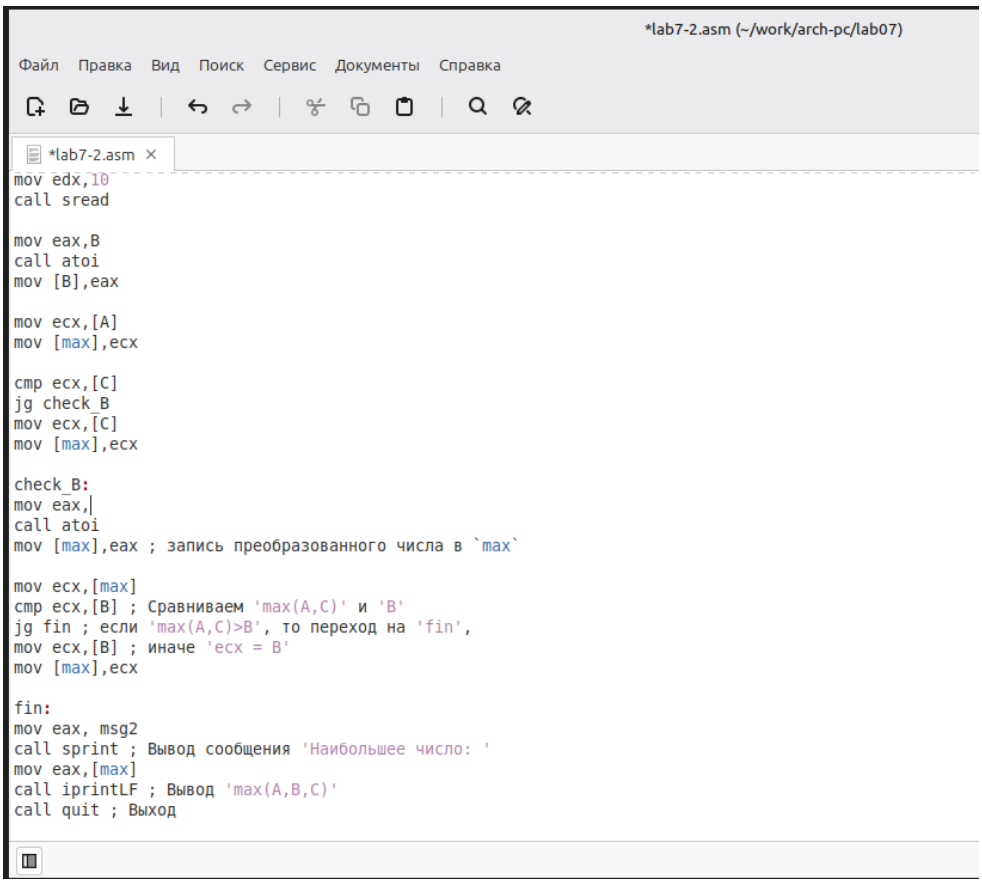

Рисунок 11: Удаление операнды в программе

Выполнила трансляцию с получением файла листинга (рис.12).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:34: error: invalid combination of opcode and operands
```

Рисунок 12: Трансляция вывода полученного файла

В новом файле листинга показывает ошибку, которая возникла при попытке трансляции файла. Никакие выходные файлы при этом помимо файла листинга не создаются. (рис. 13).

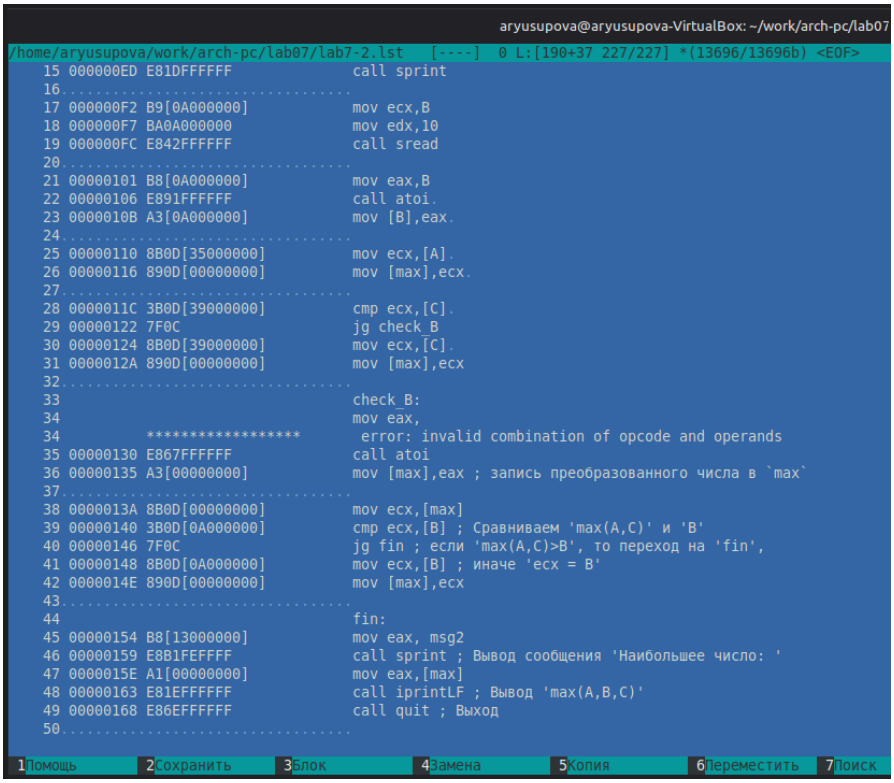

Рисунок 13: Просмотр ошибки в файле листинга

4.3 Задания для самостоятельной работы

1 задание:

Создаю необходимый файл lab2-3.asm в каталоге ~/work/arch-pc/lab07 (рис. 14).

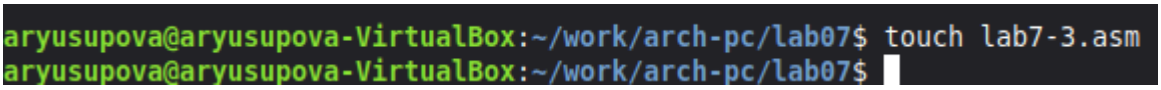

Рисунок 14: Создание файла lab7-3.asm

Открываю созданный файл и ввожу код для нахождения наименьшего значения из трёх переменных a,b,c

(рис. 15).

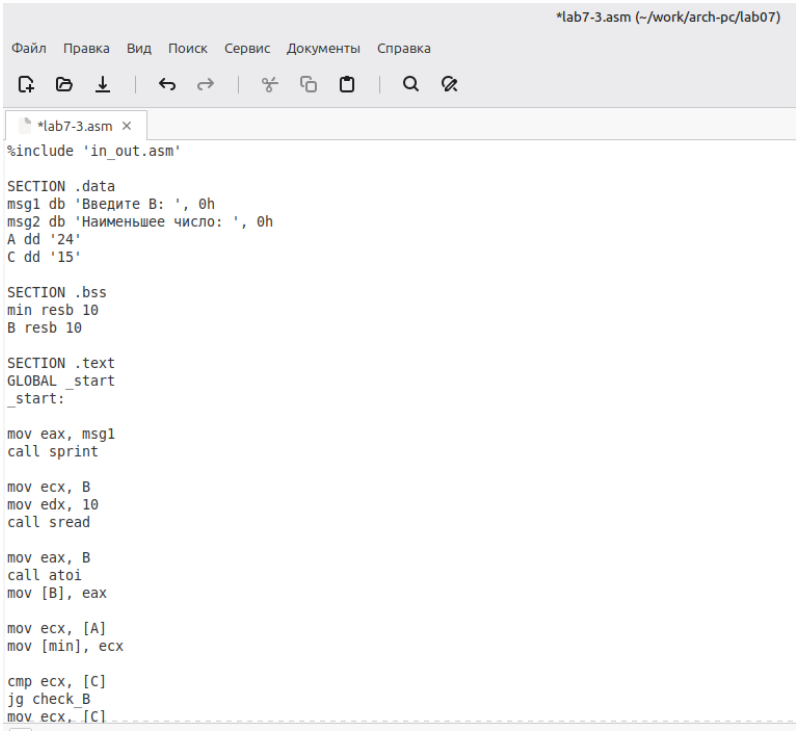

Рисунок 15: Ввод кода программы файла lab7-3.asm

Код программы 1 задание:

```
%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
```

Транслирую, компоную файл и проверяю его работу, вводя значения из таблицы вариант №10. Программа работает корректно: выводит наименьшее

значение (рис. 16).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 41
Наименьшее число: 15
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 62
Наименьшее число: 15
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 35
Наименьшее число: 15
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 16: Трансляция, компоновка файла lab7-3.asm и проверка его работы

2 задание:

Создаю необходимый файл lab7-4.asm в каталоге ~/work/arch-pc/lab07 (рис. 17).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$
```

Рисунок 17: Создание файла lab7-4.asm

Пишу программу, которая будет вычислять значение заданной функции согласно моему варианту (вариант №9) для введенных с клавиатурых переменных a и x (рис. 18).

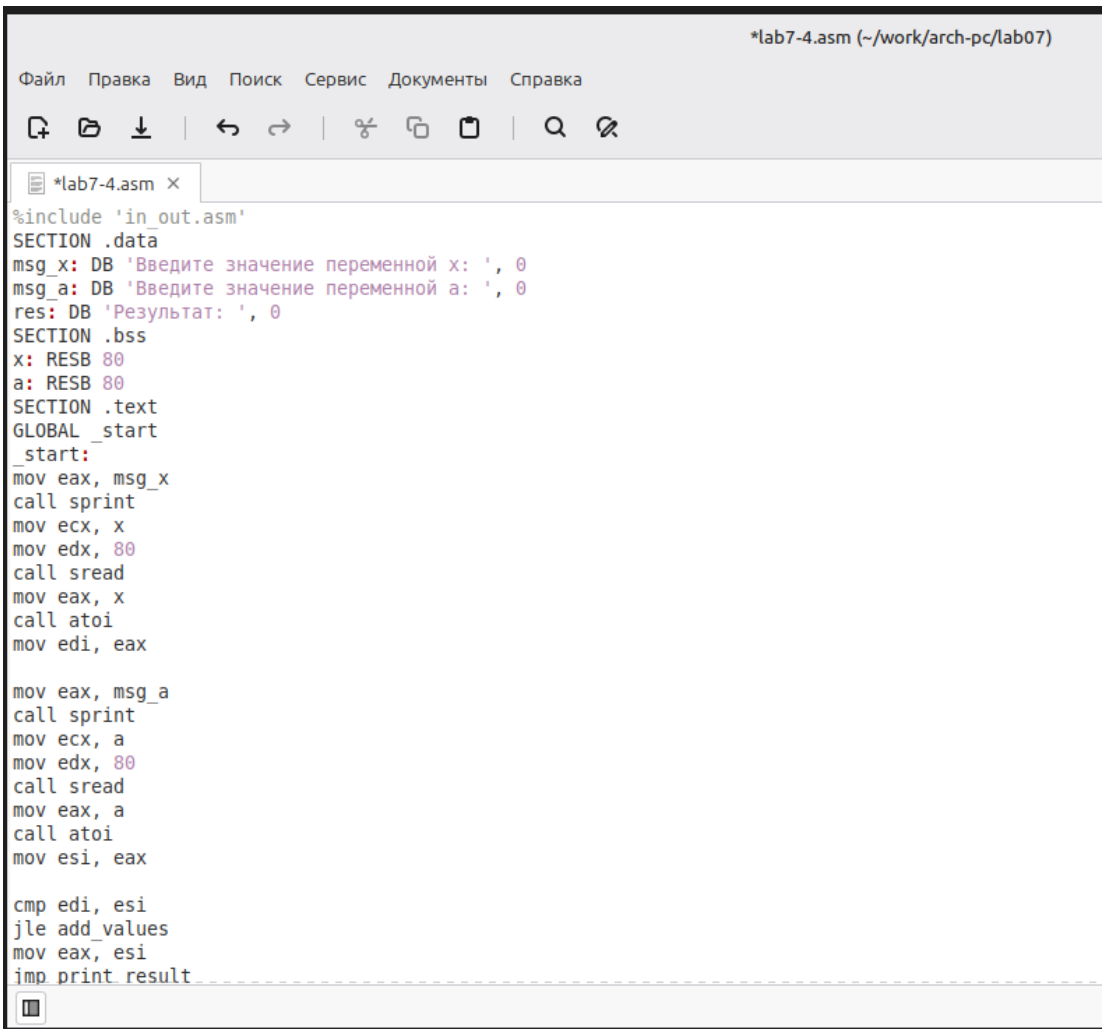

Рисунок 18:Ввод кода программы lab7-4.asm

Код программы 2 задание :

```
%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
```

Транслирую и компоную файл, запускаю и проверяю работу программы для различных значений a и x (рис. 19). В моей случае (5,7) и (6,4). Программа работает корректно.

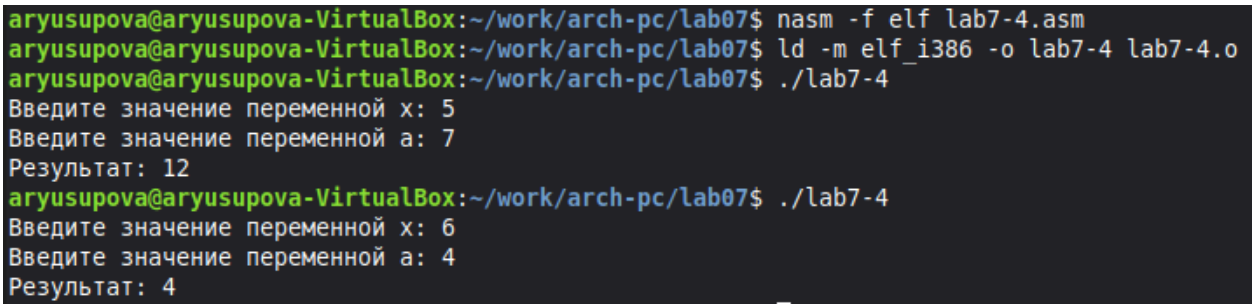

Рисунок 19: Трансляция, компоновка и запуск программы lab7-4

5 Выводы

В процессе выполнения лабораторной работы я изучила команды условных и безусловных переходов, а также приобрела навыки написания программ с их использованием. Кроме того, я ознакомилась с назначением и структурой файлов листинга.

Список литературы

1. https://esystem.rudn.ru/mod/resource/view.php?id=1030555
2. https://www.opennet.ru/docs/RUS/nasm/
3. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.