

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Юсупова Алина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1.	Цель работы.....	3
2.	Задания.....	4
3.	Теоретическое введение.....	5
4.	Выполнение лабораторной работы.....	6
4.1	Реализация переходов в NASM.....	6
4.2	Изучение структуры файла листинга.....	8
4.3	Выполнение заданий для самостоятельной работы.....	11
5.	Выводы.....	18
6.	Список литературы.....	19

1 Цель работы

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

2 Задания

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 программы без каких-либо условий.

4 Выполнение лабораторной работы.

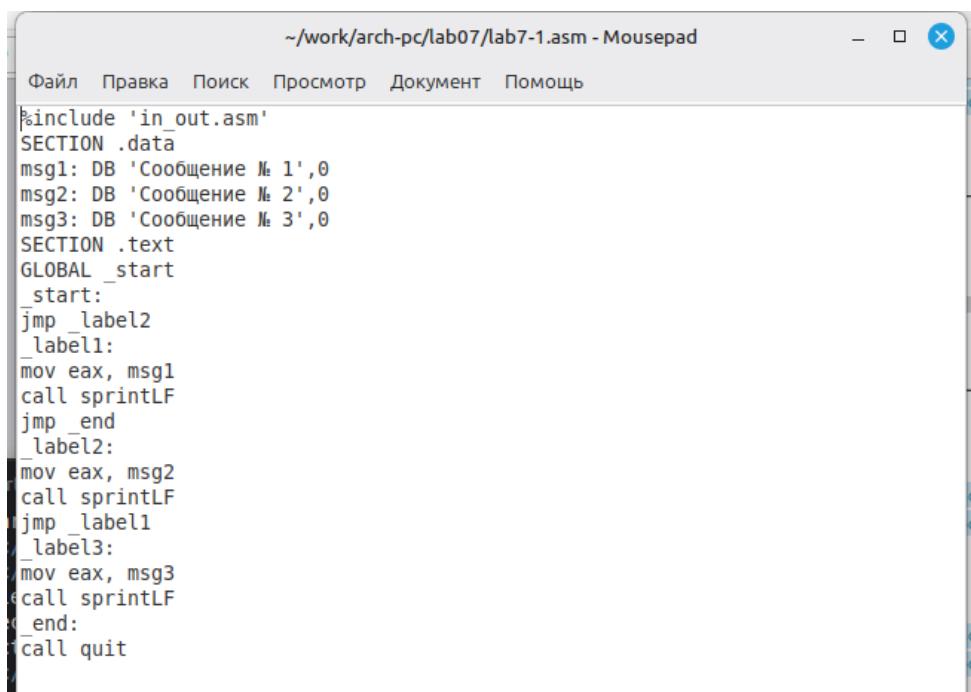
4.1 Реализация переходов в NASM.

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$
```

Рис.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).



```
~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'
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
    call sprintLF
    jmp _end
_label2:
    mov eax, msg2
    call sprintLF
    jmp _label1
_label3:
    mov eax, msg3
    call sprintLF
_end:
    call quit
```

Рис.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$ mousepad lab7-1.asm
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 № 1
```

Рис.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).

```

~/work/arch-pc/lab07/lab7-1.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%include 'in_out.asm'
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 sprintLF ; 'Сообщение № 1'
    jmp _end
_label2:
    mov eax, msg2 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 2'
    jmp _label1
_label3:
    mov eax, msg3 ; Вывод на экран строки
    call sprintLF ; 'Сообщение № 3'
    jmp _label2
_end:
    call quit ; вызов подпрограммы завершения

```

Рис.4. Изменение программы.

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

```

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 № 3
Сообщение № 2
Сообщение № 1

```

Рис.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$ 

```

Рис.6. Создание файла lab7-2.asm.

The screenshot shows a Microsoft Visual Studio Code interface with the following details:

- File Menu:** Файл, Правка, Вид, Поиск, Сервис, Документы, Справка
- Toolbar:** Includes icons for file operations (New, Open, Save, Print, Find, Replace, Copy, Paste, Cut, Undo, Redo), search, and refresh.
- Code Editor:** The current file is *lab7-2.asm. The code is as follows:

```
%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'
mov [max],ecx ; 'max = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jg check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [max],ecx ; 'max = C'
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
```

Рис.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
```

Рис.8. Проверка программы на нахождения максимума из листинга.

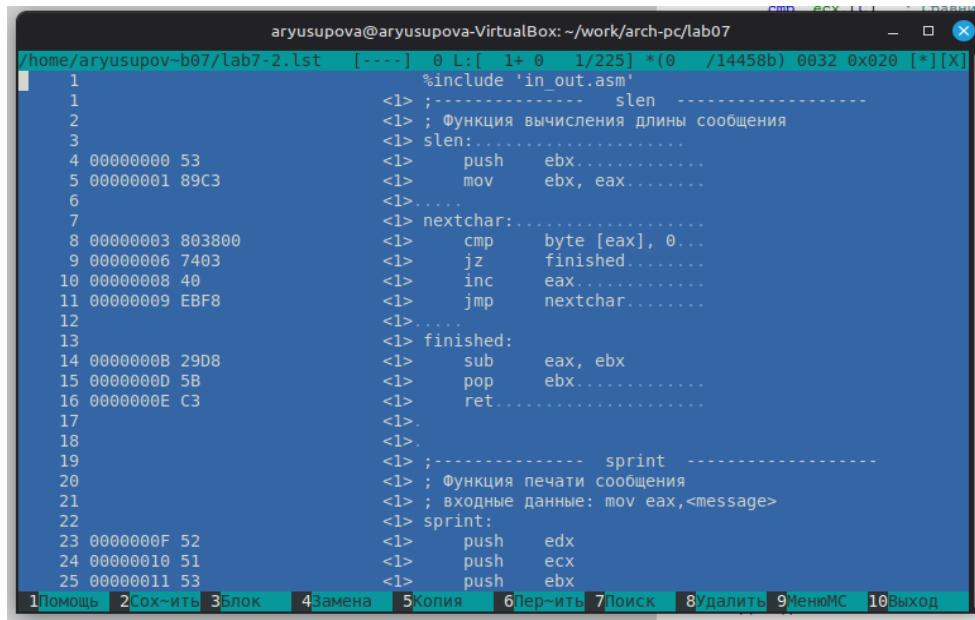
4.2 Изучение структуры файла листинга

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
```

Рис.9. Создание файла и его открытие.

Проверяю файл листинга (рис.10).



The screenshot shows a terminal window with the following text:

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07
/home/aryusupova/b07/lab7-2.lst  [---]  0 L:[ 1+ 0  1/225] *(0  /14458b) 0032 0x020 [*] [X]
1                                     %include 'in_out.asm'
1                                     ;----- slen -----
2                                     ; Функция вычисления длины сообщения
3                                     slen:.....
4 00000000 53                      push    ebx.....
5 00000001 89C3                   mov     ebx, eax.....
6                                     .....
7                                     nextchar:.....
8 00000003 803B00                 cmp     byte [eax], 0...
9 00000006 7403                   jz      finished.....
10 00000008 40                   inc     eax.....
11 00000009 EBF8                   jmp     nextchar.....
12                                     .....
13                                     finished:.....
14 0000000B 29D8                   sub     eax, ebx.....
15 0000000D 5B                   pop     ebx.....
16 0000000E C3                   ret.....
17                                     .....
18                                     .....
19                                     ;----- sprint -----
20                                     ; Функция печати сообщения
21                                     ; входные данные: mov eax,<message>
22                                     sprint:.....
23 0000000F 52                      push    edx.....
24 00000010 51                      push    ecx.....
25 00000011 53                      push    ebx.....
```

Рис.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 operand из случайной инструкции, чтобы проверить поведение файла листинга в дальнейшем (рис. 11).

```
*lab7-2.asm (~/work/arch-pc/lab07)
Файл Правка Вид Поиск Сервис Документы Справка
*lab7-2.asm X
mov edx,10
call sread

mov eax,B
call atoi
mov [B],eax

mov ecx,[A]
mov [max],ecx

cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx

check_B:
mov eax,|
call atoi
mov [max],eax ; запись преобразованного числа в `max`

mov ecx,[max]
cmp ecx,[B] ; Сравниваем `max(A,C)` и `B`
jg fin ; если `max(A,C)>B`, то переход на `fin`,
mov ecx,[B] ; иначе `ecx = B`
mov [max],ecx

fin:
mov eax, msg2
call sprint ; Вывод сообщения `Наибольшее число: `
mov eax,[max]
call iprintLF ; Вывод `max(A,B,C)`
call quit ; Выход
```

Рис.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
```

Рис.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).

```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2.lst
/home/aryusupova/work/arch-pc/lab07/lab7-2.lst [---] 0 L:[190+37 227/227] *(13696/13696b) <EOF>
15 000000ED E81DFFFFF call sprint
16 ...
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ...
21 00000101 BB[0A000000] mov eax,B
22 00000106 E891FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax.
24 ...
25 00000110 8B00[35000000] mov ecx,[A].
26 00000116 890D[00000000] mov [max],ecx.
27 ...
28 0000011C 3B00[39000000] cmp ecx,[C].
29 00000122 7F0C jg check_B
30 00000124 8B00[39000000] mov ecx,[C].
31 0000012A 890D[00000000] mov [max],ecx
32 ...
33 ...
34 *****
35 00000130 E867FFFFF call atoi
36 00000135 A3[00000000] mov [max],eax ; запись преобразованного числа в `max`
37 ...
38 0000013A 8B00[00000000] mov ecx,[max]
39 00000140 3B00[0A000000] cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 00000146 7F0C jg fin ; если 'max(A,C)>B', то переход на 'fin'
41 00000148 8B00[0A000000] mov ecx,[B] ; иначе 'ecx = B'
42 0000014E 890D[00000000] mov [max],ecx
43 ...
44 ...
45 00000154 B8[13000000] mov eax, msg2
46 00000159 E8B1FFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
47 0000015E A1[00000000] mov eax,[max]
48 00000163 E81EFFFFF call iprintfF ; Вывод 'max(A,B,C)'
49 00000168 E86EFFFFF call quit ; Выход
50 ...

```

Рис.13. Просмотр ошибки в файле листинга.

4.3 Задания для самостоятельной работы

Задание №1

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

```

aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ touch lab7-3.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ 
```

Рис.14. Создание файла lab7-3.asm.

Открываю созданный файл и ввожу код для нахождения наименьшего значения из трёх переменных a,b,c (рис. 15).

```

~/work/arch-pc/lab07/lab7-3.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь

mov ecx, [A]
mov [min], ecx

cmp ecx, [C]
jg check_B
mov ecx, [C]
mov [min], ecx

check_B:
mov eax, min
call atoi
mov [min], eax

mov ecx, [min]
cmp ecx, [B]
jb fin
mov ecx, [B]
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit

```

Рис.15. Ввод кода программы файла lab7-3.asm.

Код программы задания №1:

```
%include 'in_out.asm'

SECTION .data

msg1 db 'Введите B: ', 0h
msg2 db 'Наименьшее число: ', 0h

A dd '24'
C dd '15'
```

SECTION .bss

min resb 10

B resb 10

SECTION .text

GLOBAL _start

_start:

mov eax, msg1

call sprint

mov ecx, B

mov edx, 10

call sread

mov eax, B

call atoi

mov [B], eax

mov ecx, [A]

mov [min], ecx

cmp ecx, [C]

jg check_B

mov ecx, [C]

mov [min], ecx

check_B:

mov eax, min

call atoi

mov [min], eax

mov ecx, [min]

cmp ecx, [B]

jb fin

mov ecx, [B]

mov [min], ecx

fin:

mov eax, msg2

call sprint

mov eax, [min]

```
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$
```

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

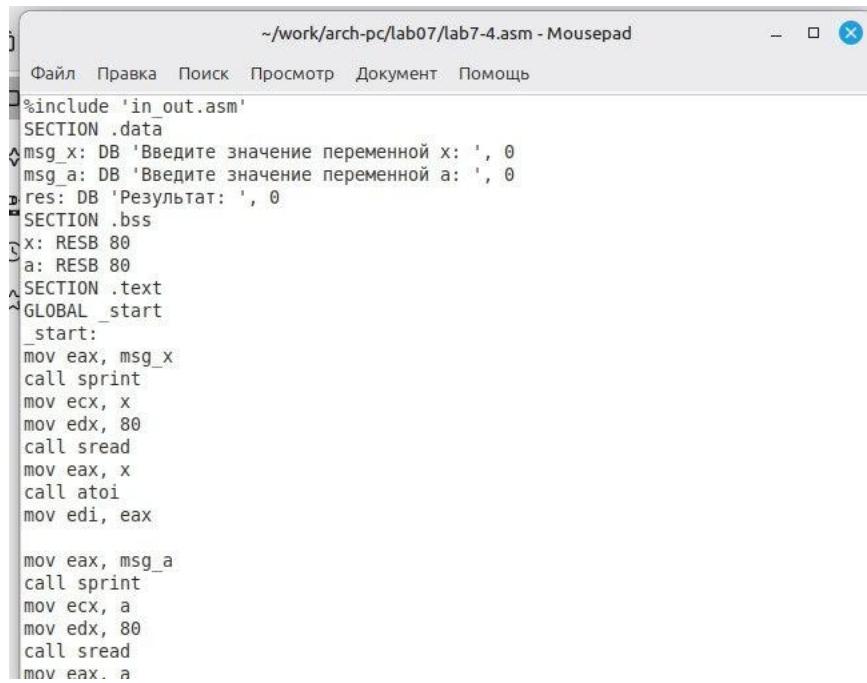
Задание №2:

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$ █
```

Рис.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).



```
~/work/arch-pc/lab07/lab7-4.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
%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
```

Рис.18. Ввод кода программы lab7-4.asm.

Код программы задания №2:

```
%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 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 а и х (рис. 19). В моем случае (5,7) и (6,4). Программа работает корректно.

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 5
Введите значение переменной a: 7
Результат: 12
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-4
Введите значение переменной x: 6
Введите значение переменной a: 4
Результат: 4
```

Рис.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.

6 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.
4. https://github.com/alyusupova/study_2025-2026_arh-pc.git