

# **Шаблон отчёта по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Хатамов Эзиз

# Содержание

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>Цель работы</b>   | <b>5</b>  |
| <b>2</b> | <b>Задание</b>   | <b>6</b>  |
| <b>3</b> | <b>Теоретическое введение</b>  | <b>7</b>  |
| <b>4</b> | <b>Выполнение лабораторной работы</b>  | <b>8</b>  |
| 4.1      | Изучение структуры файлы листинга . . . . .  | 8         |
| 4.2      | Изучение структуры файлы листинга . . . . .  | 12        |
| 4.3      | Самостоятельная работа. . . . .  | 14        |
| 4.3.1    | Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c. . . . .  | 14        |
| 4.3.2    | Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции f(x) и выводит результат вычислений. . . . . | 15        |
| <b>5</b> | <b>Выводы</b>  | <b>17</b> |
|          | <b>Список литературы</b>   | <b>18</b> |

## Список иллюстраций

|      |   |    |
|------|---|----|
| 4.1  | Создания каталога и файла . . . . .   | 8  |
| 4.2  | Скопирования файла in_out.asm в нужный каталог . . . . .  | 9  |
| 4.3  | Программа с использованием инструкции jmp . . . . .   | 9  |
| 4.4  | Создания исполняемого файла . . . . .   | 10 |
| 4.5  | Изменения текста файла . . . . .  | 10 |
| 4.6  | Создания(изменённого) исполняемого файла . . . . .  | 10 |
| 4.7  | Создания файла lab7-2.asm . . . . .   | 11 |
| 4.8  | Программа, которая определяет и выводит на экран наибольшую из<br>3 целочисленных переменных: А,В и С . . . . . | 11 |
| 4.9  | Создания исполняемого файла lab7-2.asm . . . . .  | 12 |
| 4.10 | Создания листинга . . . . .   | 12 |
| 4.11 | Открытие листинга . . . . .   | 12 |
| 4.12 | 112 строка для объяснения . . . . .   | 13 |
| 4.13 | 14 строка для объяснения . . . . .  | 13 |
| 4.14 | 42 строка для объяснения . . . . .  | 13 |
| 4.15 | Ошибка в программе . . . . .  | 13 |
| 4.16 | Осмотр листинга . . . . .   | 14 |
| 4.17 | Внесения программы в файл . . . . .   | 14 |
| 4.18 | Создания исполняемого файла lab7-3.asm . . . . .  | 15 |
| 4.19 | Внесения программы в файл lab7-4.asm . . . . .  | 15 |
| 4.20 | Создания исполняемого файла lab7-4.asm . . . . .  | 16 |

## **Список таблиц**

# 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

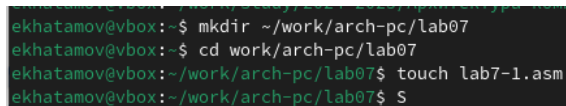
### **3 Теоретическое введение**

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 Изучение структуры файлы листинга

Для начала я создал каталог для программ Лабораторной работы. потом перешёл в него и создал файл lab07-1.asm (рис. 4.1).



```
ekhatamov@vbox:~$ mkdir ~/work/arch-pc/lab07
ekhatamov@vbox:~$ cd work/arch-pc/lab07
ekhatamov@vbox:~/work/arch-pc/lab07$ touch lab7-1.asm
ekhatamov@vbox:~/work/arch-pc/lab07$ S
```

Рис. 4.1: Создания каталога и файла

Потом зашел на МС и через него скопировал файл in\_out.asm в созданный каталог



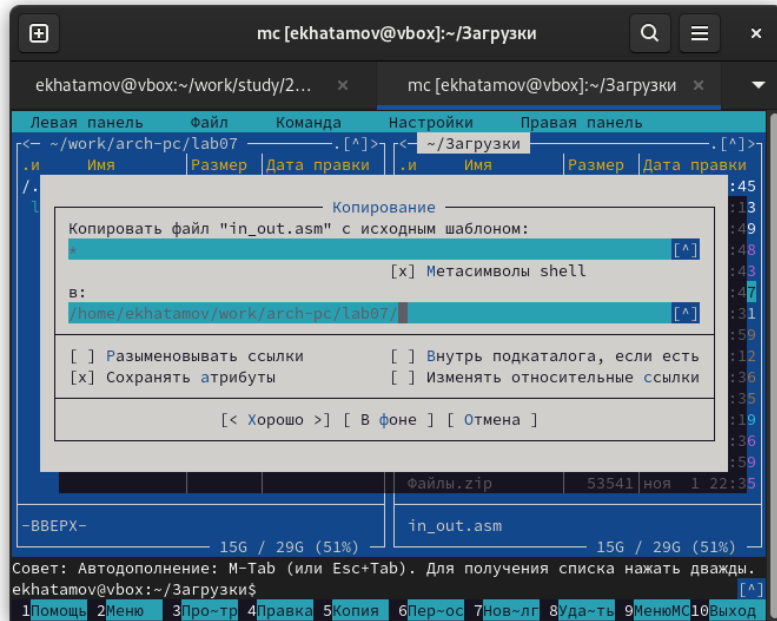


Рис. 4.2: Скопирования файла in\_out.asm в нужный каталог

После этого я открыл созданной мною файл с помощью клавиши F4 и ввел туда программу с использованием инструкции jmp

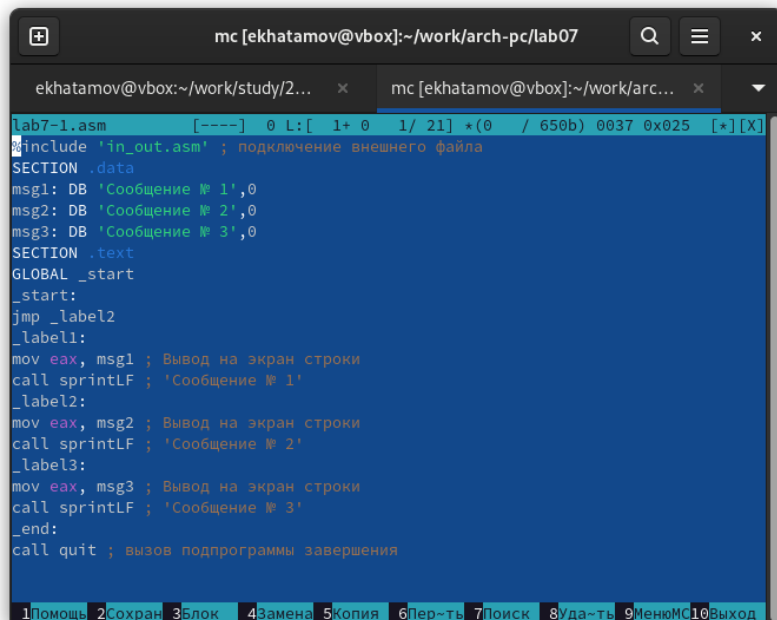


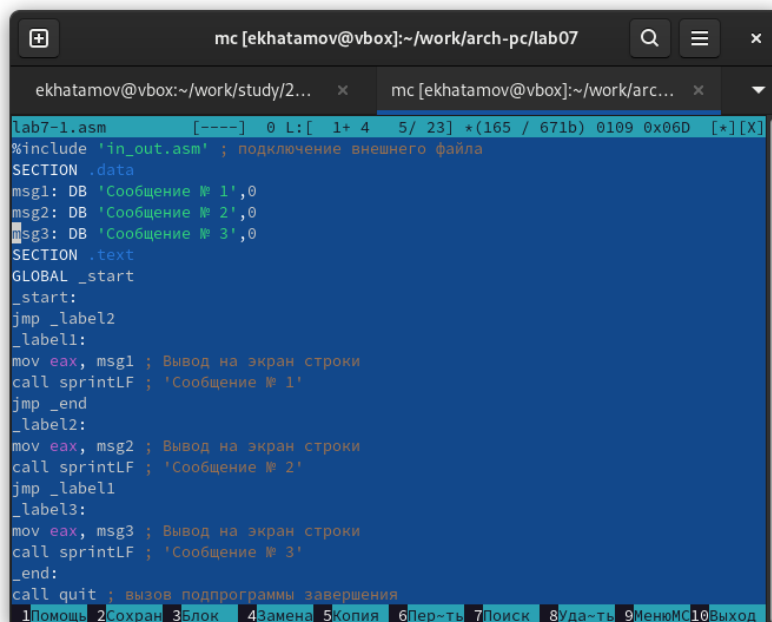
Рис. 4.3: Программа с использованием инструкции jmp

Потом я создал исполняемый файл и запустил его

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

Рис. 4.4: Создания исполняемого файла

Я изменил текст файла чтобы осуществить переход назад в инструкции `jmp`. Для этого в текст программы после вывода сообщения № 2 добавил инструкцию `jmp` с меткой `_label1`, и после вывода сообщения № 1 добавил инструкцию `jmp` с меткой `_end`



```
lab7-1.asm [----] 0 L: [ 1+ 4 5/ 23] *(165 / 671b) 0109 0x06D [*][X]
#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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8Уда-ть 9Меню 10Выход
```

Рис. 4.5: Изменения текста файла

Создал исполняемый файл и запустил его ещё раз но уже изменённого.

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

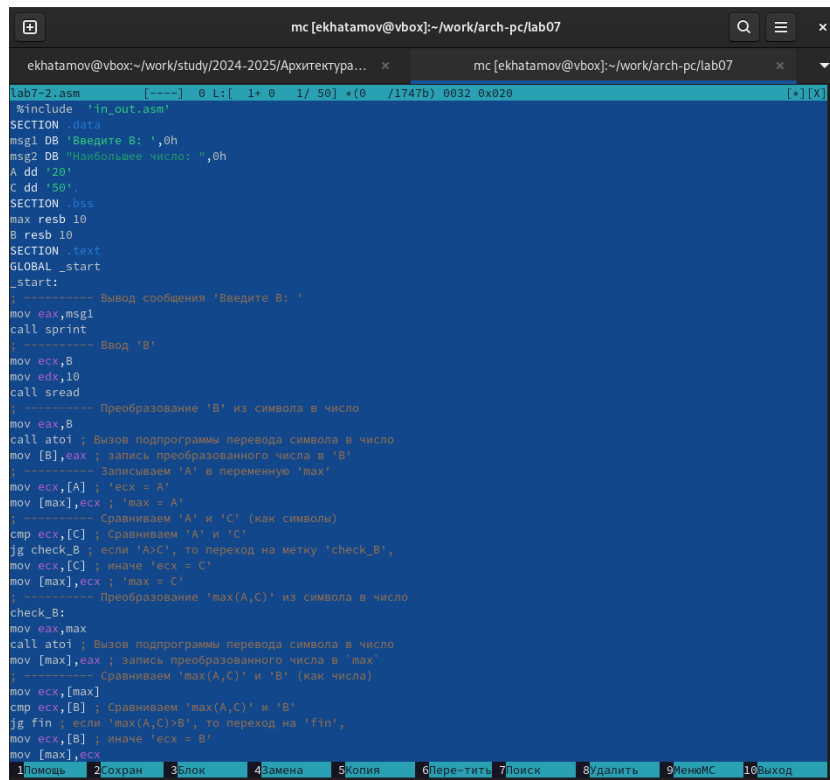
Рис. 4.6: Создания(изменённого) исполняемого файла

Потом я создал новый файл в том же каталоге lab7-2.asm

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

Рис. 4.7: Создания файла lab7-2.asm

После создания я открыл файл и ввёл туда программу, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C



```
lab7-2.asm [---] 0 L: 1+ 0 1/ 50) *(0 /1747b) 0032 0x020 [*] [X]
%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:
mov eax,max
call atoi ; Вызов подпрограммы перевода символа в число
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
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
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пере-титы 7Поиск 8Удалить 9МенюМС 10Выход
```

Рис. 4.8: Программа, которая определяет и выводит на экран наибольшую из 3 целочисленных переменных: A, B и C

Потом создал исполняемый файл и запустил его. И ещё я проверил его работу

```
ekhatamov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
ekhatamov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
ekhatamov@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 25
Наибольшее число: 50
ekhatamov@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 20
Наибольшее число: 50
ekhatamov@vbox:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 60
Наибольшее число: 60
ekhatamov@vbox:~/work/arch-pc/lab07$
```

Рис. 4.9: Создания исполняемого файла lab7-2.asm

## 4.2 Изучение структуры файлы листинга

Я создал файл листинга с помощью nasm указав ключ -l и задал имя лисинга в командной строке

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

Рис. 4.10: Создания листинга

Потом октрыл файл листинга с помощью mcedit и изучил содержимое

```
lab7-2.lst  [-M--] 63 L: [ 1+ 0 1/225] *(63 /14461b) 0010 0x00A [*][X]
1                                     %include 'in_out.asm'
2                                     <1> ;----- slen -----
3                                     <1> ; Функция вычисления длины сообщения
4                                     <1> slen:.....
5                                     <1>   push    ebx.....
6                                     <1>   mov     ebx, eax.....
7                                     <1>.....
8                                     <1> nextchar:.....
9                                     <1>   cmp     byte [eax], 0...
10                                    <1>   jz      finished.....
11                                    <1>   inc     eax.....
12                                    <1>   jmp     nextchar.....
13                                    <1>.....
14                                    <1> finished:
15                                    <1>   sub     eax, ebx
16                                    <1>   pop     ebx.....
17                                    <1>   ret.....
18                                    <1>.....
19                                    <1> ;----- sprint -----
20                                    <1> ; Функция печати сообщения
21                                    <1> ; входные данные: mov eax,<message>
```

Рис. 4.11: Открытие листинга

Выбрал первую строку и это 112. В строке которая показана в картинке снизу обозначается “00000086” — адрес в памяти, “E8C9FFFFFF” — машинный код для инструкции call а “call inprint” — обозначает вызов функции inprint.



Рис. 4.12: 112 строка для объяснения

Выбрал вторую строку и это 14. В строке которая показана в картинке снизу обозначается “0000000B” — адрес в памяти, где расположена эта инструкция, 29D8 — машинный код для инструкции sub а “sub eax, ebx” — обозначает операцию, которая вычитает значение регистра ebx из значения регистра eax и сохраняет результат в eax.

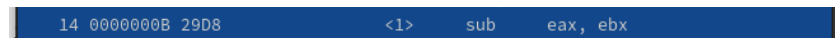


Рис. 4.13: 14 строка для объяснения

Выбрал третью строку и это 42. В строке которая показана в картинке снизу обозначается “00000153” — адрес в памяти, где расположена эта инструкция, 890D — машинный код для инструкции mov а “mov [max], ecx” — Обозначает операцию, которая копирует значение из регистра ecx в память по адресу, соответствующему метке или переменной max.



Рис. 4.14: 42 строка для объяснения

Потом в строке mov eax,max я убрал max и попробовал создать файл. Выдало ошибку, так как для программы нужно два операнда.

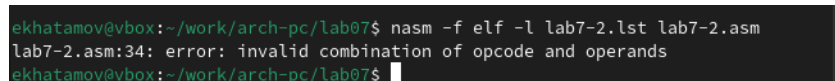


Рис. 4.15: Ошибка в программе

В файле листинга показывает где ошибка и с чем оно связана

```

34          ***** error: invalid combination of opcode an
d operands
35 00000130 E867FFFFFF call atoi ; Вызов подпрограммы перевода

```

Рис. 4.16: Осмотр листинга

## 4.3 Самостоятельная работа.

### 4.3.1 Напишите программу нахождения наименьшей из 3 целочисленных переменных a,b и c.

Для начала я создал файл и в него я написал программу нахождения наименьшей из 3 целочисленных переменных a,b и c.

```

/home/ekhatamov/work/arch-pc/lab07/lab7-3.asm 1527/1748 87%
#include 'in_out.asm'
SECTION .data
msg1 DB 'Введите B: ',0h
msg2 DB "Наименьшее число: ",0h
A dd '8'
C dd '68'
SECTION .bss
min 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 'min'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ; 'min = A'
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A < C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = C'
mov [min],ecx ; 'min = C'
; ----- Преобразование 'min(A,C)' из символа в число
check_B:
mov eax,min
call atoi ; Вызов подпрограммы перевода символа в число
mov [min],eax ; запись преобразованного числа в 'min'
; ----- Сравниваем 'min(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B] ; Сравниваем 'min(A,C)' и 'B'
jl fin ; если 'min(A,C) < B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [min],ecx
1Помощь 2Заверн 3Выход 4Лех 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход

```

Рис. 4.17: Внесения программы в файл

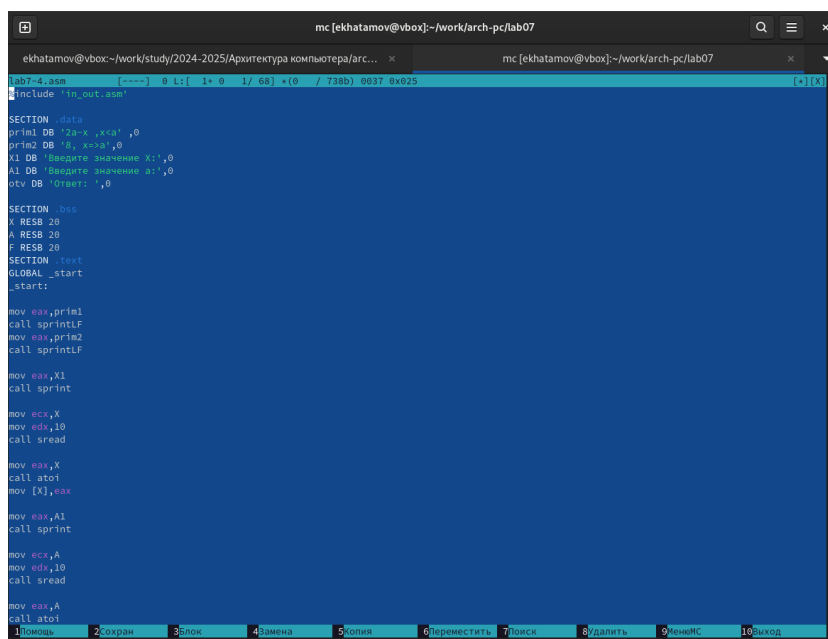
Потом создал исполняемый файл и запустил его и проверил все ли работает

```
ekhatamov@vbox: ~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
ekhatamov@vbox: ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
ekhatamov@vbox: ~/work/arch-pc/lab07$ ./lab7-3
Введите В: 88
Наименьшее число: 8
ekhatamov@vbox: ~/work/arch-pc/lab07$
```

Рис. 4.18: Создания исполняемого файла lab7-3.asm

### 4.3.2 Напишите программу, которая для введенных с клавиатуры значений $x$ и $a$ вычисляет значение заданной функции $f(x)$ и выводит результат вычислений.

Для начала я создал файл и в него я написал программу, которая для введенных с клавиатуры значений  $x$  и  $a$  вычисляет значение заданной функции  $f(x)$  и выводит результат вычислений.



```
lab7-4.asm 0 L: 1* 0 1/ 68 *0 / 7385 0637 0x025
include "in_out.asm"

SECTION .data
prtm1 DB "2a=x , x^a" , 0
prtm2 DB "a , x->a" , 0
X1 DB "Введите значение X:" , 0
A1 DB "Введите значение a:" , 0
str DB "Output: " , 0

SECTION .bss
X RESB 20
A RESB 20
F RESB 20

SECTION .text
GLOBAL _start
_start:

mov eax, prtm1
call sprintf
mov eax, prtm2
call sprintf

mov eax, X1
call sprintf

mov ecx, X
mov edx, 10
call read

mov eax, X
call atoi
mov [X], eax

mov eax, A1
call sprintf

mov ecx, A
mov edx, 10
call read

mov eax, A
call atoi
```

Рис. 4.19: Внесения программы в файл lab7-4.asm

После этого я создал исполняемый файл и запустил его. потом я написал цифры которые были таблице на  $X$  и на  $A$

```
ekhatamov@vbox:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
ekhatamov@vbox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
ekhatamov@vbox:~/work/arch-pc/lab07$ ./lab7-4
2a-x ,x<a
8, x=>a
Введите значение X:1
Введите значение a:2
Ответ: 3
ekhatamov@vbox:~/work/arch-pc/lab07$ ./lab7-4
2a-x ,x<a
8, x=>a
Введите значение X:2
Введите значение a:1
Ответ: 8
ekhatamov@vbox:~/work/arch-pc/lab07$
```

Рис. 4.20: Создания исполняемого файла lab7-4.asm

Все готова!



## 5 Выводы

Я изучил команды условного и безусловного перехода. Приобрел навыки написания программ с переходами.

## **Список литературы**

(<https://esystem.rudn.ru>) Архитектура компьютеров, Лабораторная работа №7