

Очѐт по лабораторной работе № 8

Нкнбд-05-2023

Диого Элизеу Луиж Музумбо

Содержание

1	Цель работы	4
2	Задание	5
3	Выполнение лабораторной работы	6
3.1	Реализация переходов в NASM	6
3.2	Изучение структуры файлы листинга	11
3.3	Задание для самостоятельной работы	13
4	Выводы	15

Список иллюстраций

3.1	Название рисунка	6
3.2	Название рисунка	7
3.3	Название рисунка	7
3.4	Название рисунка	8
3.5	Название рисунка	9
3.6	Название рисунка	9
3.7	Название рисунка	9
3.8	Название рисунка	10
3.9	Название рисунка	10
3.10	Название рисунка	11
3.11	Название рисунка	11
3.12	Название рисунка	11
3.13	Название рисунка	12
3.14	Название рисунка	12
3.15	Название рисунка	12
3.16	Название рисунка	13
3.17	Название рисунка	13
3.18	Название рисунка	14
3.19	Название рисунка	14

1 Цель работы

Изучить команды условного и безусловного переходов. Приобрести навыков написания программ с использованием переходов. Ознакомиться с назначением и структурой файла листинга.

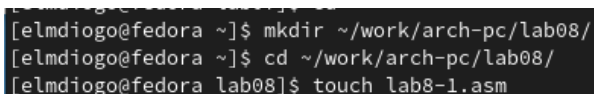
2 Задание

1. Реализовать переходы в NASM
2. Изучить структуру файлов листинга
3. Выполнить задание для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация переходов в NASM

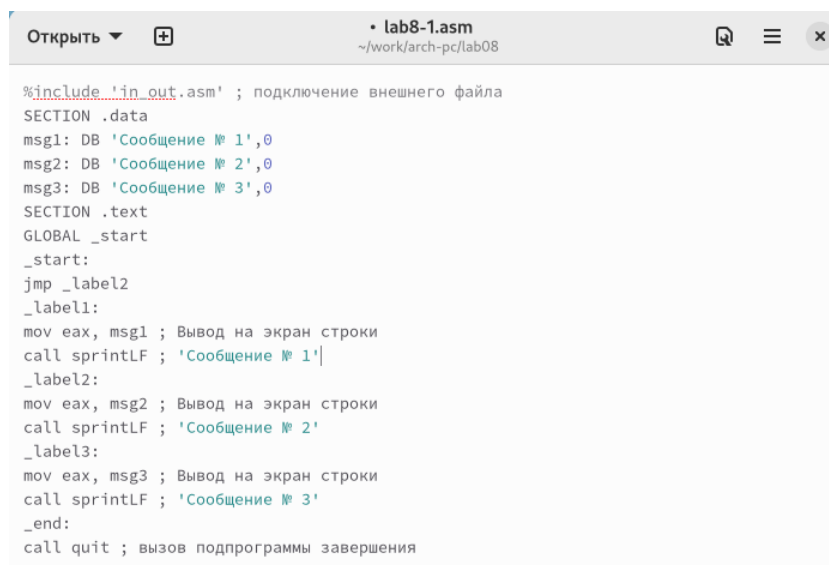
1. Создали каталог для программам лабораторной работы № 8, перешли в него и создали файл lab8-1.asm: (рис. 3.1)



```
[elmdiego@fedora ~]$ mkdir ~/work/arch-pc/lab08/  
[elmdiego@fedora ~]$ cd ~/work/arch-pc/lab08/  
[elmdiego@fedora lab08]$ touch lab8-1.asm
```

Рис. 3.1: Название рисунка

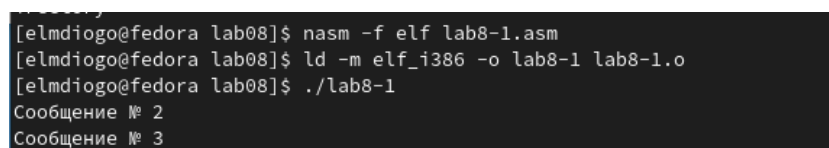
2. Инструкция `jmp` в NASM используется для реализации безусловных переходов. Рассмотрели пример программы с использованием инструкции `jmp`. Ввели в файл lab8-1.asm текст программы из листинга 8.1. (рис. 3.2)



```
%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 printf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call printf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call printf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.2: Название рисунка

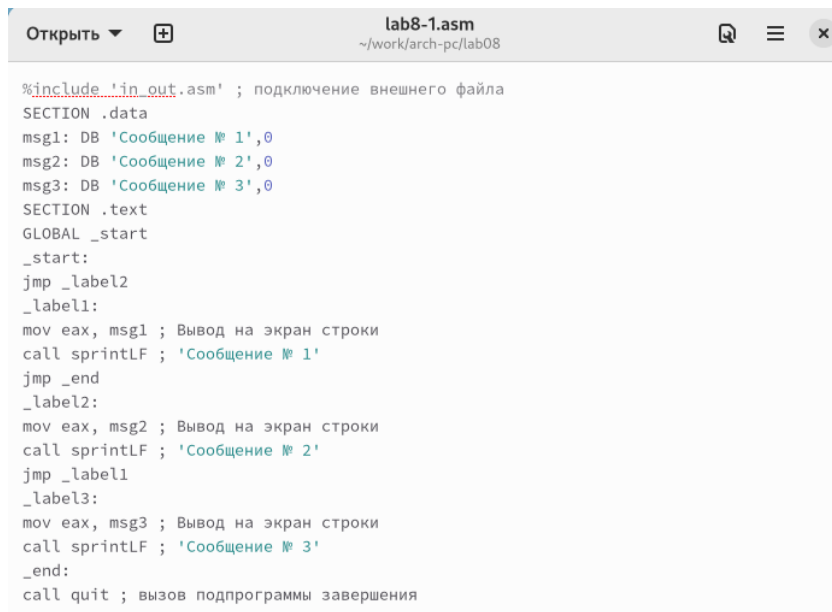
Создали исполняемый файл и запустили его. Результат работы данной программы следующий: (рис. 3.3)



```
[elmdigo@fedora lab08]$ nasm -f elf lab8-1.asm
[elmdigo@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[elmdigo@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 3
```

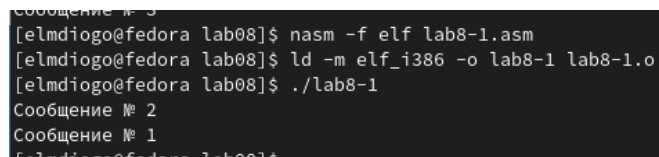
Рис. 3.3: Название рисунка

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения. Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменили программу таким образом, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавили инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавили инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`). Изменили текст программы в соответствии с листингом 8.2 (рис. ??), (рис. 3.4)



```
Открыть + lab8-1.asm ~/work/arch-pc/lab08

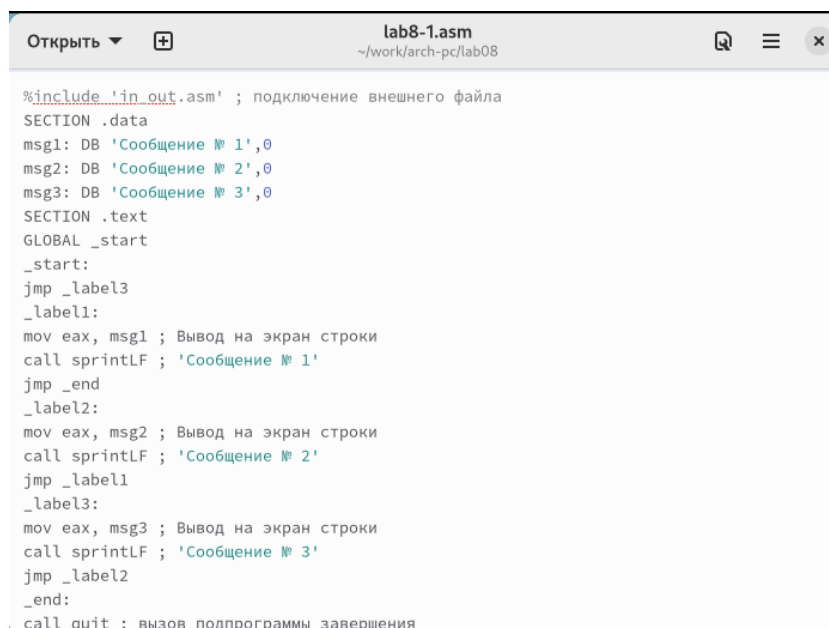
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2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



```
Сообщение № 3
[elmdiago@fedora lab08]$ nasm -f elf lab8-1.asm
[elmdiago@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[elmdiago@fedora lab08]$ ./lab8-1
Сообщение № 2
Сообщение № 1
[elmdiago@fedora lab08]$
```

Рис. 3.4: Название рисунка

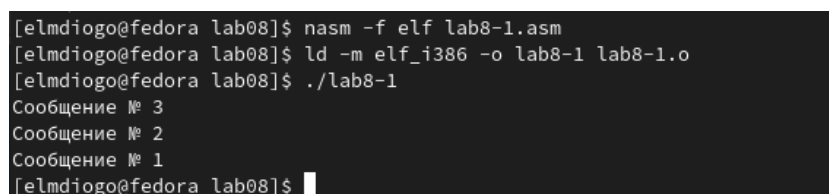
Измените текст программы добавив или изменив инструкции `jmp`. (рис. 3.5), (рис. 3.6)



```
Открыть ▾ + lab8-1.asm
~/work/arch-pc/lab08

%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
```

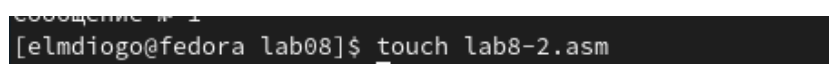
Рис. 3.5: Название рисунка



```
[elmdigo@fedora lab08]$ nasm -f elf lab8-1.asm
[elmdigo@fedora lab08]$ ld -m elf_i386 -o lab8-1 lab8-1.o
[elmdigo@fedora lab08]$ ./lab8-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[elmdigo@fedora lab08]$
```

Рис. 3.6: Название рисунка

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрели 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. Создали файл `lab8-2.asm` в каталоге `~/work/arch-pc/lab08`. (рис. 3.7) Внимательно изучили текст программы из листинга 8.3 и ввели в `lab8-2.asm`. (рис. 3.8)



```
[elmdigo@fedora lab08]$ touch lab8-2.asm
```

Рис. 3.7: Название рисунка

```

Открыть  + lab8-2.asm
~/work/arch-pc/lab08

%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'

```

Рис. 3.8: Название рисунка

Создали исполняемый файл и проверили его работу для разных значений B.
(рис. 3.9)

```

[elmdigo@fedora lab08]$ nasm -f elf lab8-2.asm
[elmdigo@fedora lab08]$ ld -m elf_i386 -o lab8-2 lab8-2.o
[elmdigo@fedora lab08]$ ./lab8-2
Введите B: 5
Наибольшее число: 50
[elmdigo@fedora lab08]$ ./lab8-2
Введите B: 100
Наибольшее число: 100

```

Рис. 3.9: Название рисунка

Обратили внимание, в данном примере переменные A и C сравниваются как символы, а переменная B и максимум из A и C как числа (для этого используется функция `atoi` преобразования символа в число). Это сделано для демонстрации того, как сравниваются данные. Данную программу можно упростить и сравнивать все 3 переменные как символы (т.е. не использовать функцию `atoi`). Однако если переменные преобразовать из символов числа, над ними можно корректно проводить арифметические операции.

3.2 Изучение структуры файлы листинга

4. Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке. Создали файл листинга для программы из файла lab8-2.asm. (рис. 3.10)

```
[elmdio@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
```

Рис. 3.10: Название рисунка

Открыли файл листинга lab8-2.lst с помощью текстового редактора mcedit: (рис. 3.11), (рис. 3.12)

```
[elmdio@fedora lab08]$ mcedit lab8-2.lst
```

Рис. 3.11: Название рисунка

```
lab8-2.lst [----] 0 L: [ 13+ 0 13/225] *(850 /14458b) 0032 0x020 [*][X]
13                                     <1>.....
14                                     <1> finished:
15 0000000B 29D8                       <1>    sub    eax, ebx
16 0000000D 5B                         <1>    pop    ebx.....
17 0000000E C3                         <1>    ret.....
18                                     <1>.
19                                     <1>.
20                                     <1> ;----- sprint -----
21                                     <1> ; Функция печати сообщения
22                                     <1> ; входные данные: mov eax,<message>
23                                     <1> sprint:
24 0000000F 52                         <1>    push   edx
25 00000010 51                         <1>    push   ecx
26 00000011 53                         <1>    push   ebx
27 00000012 50                         <1>    push   eax
28 00000013 E8E8FFFFFF                 <1>    call   slen
29                                     <1>.....
30 00000018 89C2                       <1>    mov    edx, eax
31 0000001A 58                         <1>    pop    eax
32                                     <1>.....
33 0000001B 89C1                       <1>    mov    ecx, eax
34 0000001D BB01000000                 <1>    mov    ebx, 1

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9Меню10Выход
```

Рис. 3.12: Название рисунка

Внимательно ознакомились с его форматом и содержимым. Содержимое трёх строк файла листинга: 1)45 00000154 B8[13000000] mov eax, msg2 - строка 45,

адрес 00000154, B8[13000000] - машинный код, mov eax, msg2 - исходный текст программы 2)46 00000159 E8B1FEFFFF call sprint - строка 46, адрес 00000159, E8B1FEFFFF - машинный код, call sprint - исходный текст программы 3)47 0000015E A1[00000000] mov eax,[max] - строка 47, адрес 0000015E, A1[00000000] - машинный код, mov eax,[max] - исходный текст программы

Открыли файл с программой lab8-2.asm и в инструкции mov с двумя операндами удалить один операнд. (рис. 3.13) Выполните трансляцию с получением файла листинга: (рис. 3.14), (рис. 3.15)

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

Рис. 3.13: Название рисунка

```
[elmdingo@fedora lab08]$ nasm -f elf -l lab8-2.lst lab8-2.asm
```

Рис. 3.14: Название рисунка

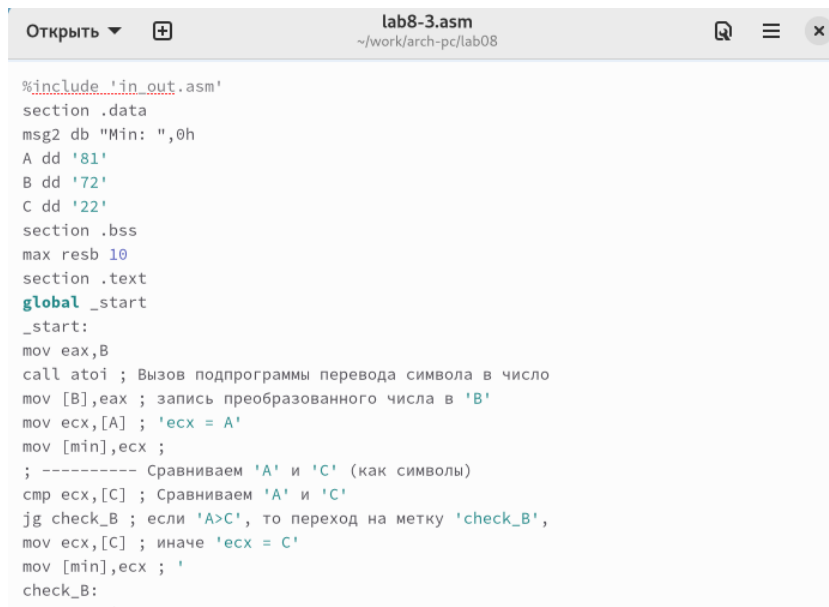
```
34 00000130 B8[00000000]    mov eax,max
35 00000135 E862FEFFFF    call atoi ; Вызов подпрограммы перевода
36 0000013A A3[00000000]    mov [max],eax ; запись преобразованного
37                      ; ----- Сравниваем 'max(A,C)' и 'B'
38 0000013F 8B0D[00000000]    mov ecx,[max]
39 00000145 3B0D[0A000000]    cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 0000014B 7F0C          jg fin ; если 'max(A,C)>B', то переход н
41 0000014D 8B0D[0A000000]    mov ecx,[B] ; иначе 'ecx = B'
42 00000153 890D[00000000]    mov [max],ecx
43                      ; ----- Вывод результата
44                      fin:
45 00000159 B8[13000000]    mov eax, msg2
46 0000015E E8ACFEFFFF    call sprint ; Вывод сообщения 'Наибольшее
47                      mov eax
47                      *****
48 00000163 E81EFEFFFF    call iprintLF ; Вывод 'max(A,B,C)'
49 00000168 E86EFEFFFF    call quit ; Выход
error: invalid combination of opcode and
```

Рис. 3.15: Название рисунка

Создаётся выходной файл lst. В листинге добавляется сообщение об ошибке.

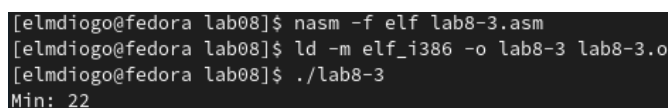
3.3 Задание для самостоятельной работы

1. Написали программу нахождения наименьшей из 3 целочисленных переменных a, b и c. (рис. 3.16) Значения переменных выбрали из таблицы в соответствии с 14 вариантом, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу. (рис. 3.17)



```
%include 'in_out.asm'
section .data
msg2 db "Min: ",0h
A dd '81'
B dd '72'
C dd '22'
section .bss
max resb 10
section .text
global _start
_start:
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'B'
mov ecx,[A] ; 'ecx = A'
mov [min],ecx ;
; ----- 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 [min],ecx ;
check_B:
mov ecx,min
```

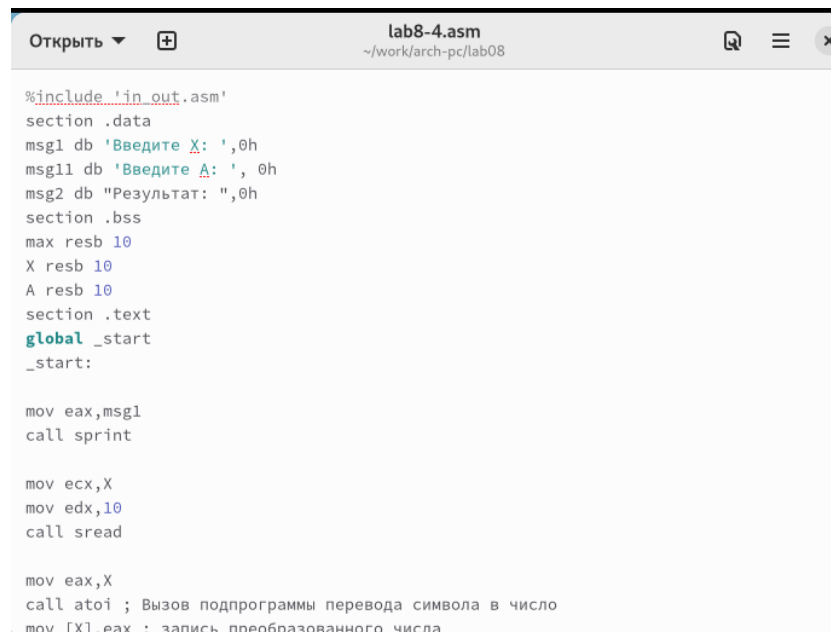
Рис. 3.16: Название рисунка



```
[elmdiago@fedora lab08]$ nasm -f elf lab8-3.asm
[elmdiago@fedora lab08]$ ld -m elf_i386 -o lab8-3 lab8-3.o
[elmdiago@fedora lab08]$ ./lab8-3
Min: 22
```

Рис. 3.17: Название рисунка

2. Написали программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. (рис. 3.18) Вид функции $f(x)$ выбрали из таблицы вариантов заданий в соответствии с вариантом 14, полученным при выполнении лабораторной работы № 7. Создали исполняемый файл и проверили его работу для значений x и a. (рис. 3.19)



The screenshot shows a text editor window titled "lab8-4.asm" with the path "~/work/arch-pc/lab08". The code is as follows:

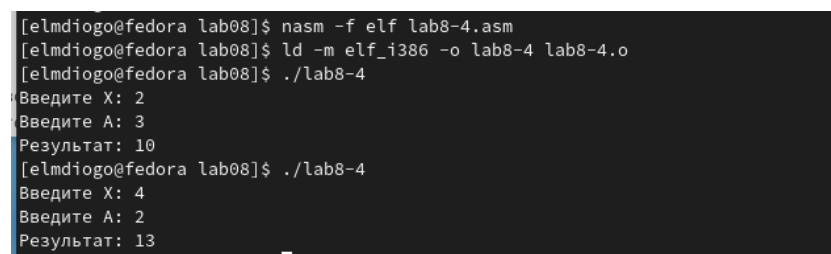
```
%include 'in_out.asm'
section .data
msg1 db 'Введите X: ',0h
msg11 db 'Введите A: ', 0h
msg2 db "Результат: ",0h
section .bss
max resb 10
X resb 10
A resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,X
mov edx,10
call sread

mov eax,X
call atoi ; Вызов подпрограммы перевода символа в число
mov [X],eax ; запись преобразованного числа
```

Рис. 3.18: Название рисунка



The screenshot shows a terminal window with the following commands and output:

```
[elmdiago@fedora lab08]$ nasm -f elf lab8-4.asm
[elmdiago@fedora lab08]$ ld -m elf_i386 -o lab8-4 lab8-4.o
[elmdiago@fedora lab08]$ ./lab8-4
Введите X: 2
Введите A: 3
Результат: 10
[elmdiago@fedora lab08]$ ./lab8-4
Введите X: 4
Введите A: 2
Результат: 13
```

Рис. 3.19: Название рисунка

4 Выводы

В ходе выполнения лабораторной работы были изучены команды условного и безусловного переходов. Были приобретены навыки написания программ с использованием переходов. Ознакомились с назначением и структурой файла листинга.