

Лабораторная работа №7

Отчет

Устинова Виктория Вадимовна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	19

Список иллюстраций

3.1	Переходим в каталог и создаем там файл lab7-1.asm	7
3.2	Заполняем данный файл	7
3.3	Смотрим как работает файл	8
3.4	Редактируем файл	8
3.5	Смотрим как работает файл	8
3.6	Редактируем данный файл	9
3.7	Сверяемся с нужным выводом, все верно	9
3.8	Используем команду touch	9
3.9	Заполняем файл как указано в листинге	10
3.10	Смотрим как работает наш файл	10
3.11	Используем Mscedit	11
3.12	Отрываем файл	11
3.13	Удалили операндум	13
3.14	Транслируем файл	14
3.15	Просматриваем ошибку в файле листинга	14
3.16	Создаем файл	14
3.17	Сама программа	15
3.18	Проверяем работу	15
3.19	Создаем новый файл	16
3.20	Пишем новую программу, для заданных нам условий	17
3.21	Проверяем выполнение, все сделано корректно	18

Список таблиц

1 Цель работы

Освоить условный и безусловный переход. Ознакомиться с назначением и структурой файла листинга.

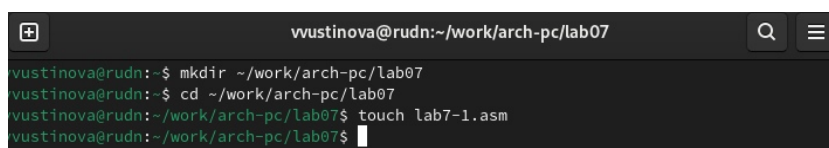
2 Задание

Написать две программы и выполнить лабораторную работу №7

3 Выполнение лабораторной работы

Реализация переходов в NASM

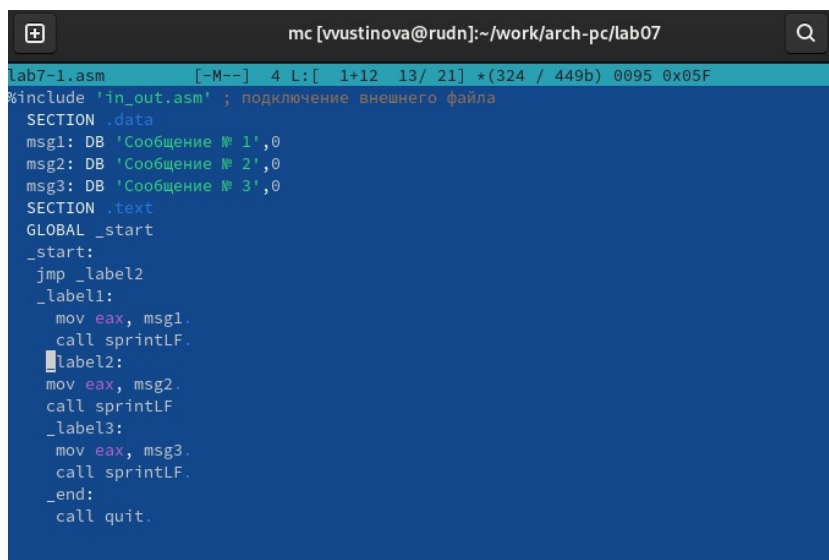
Создаем каталог для 7 лабораторной работы(рис. 3.1).



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

Рис. 3.1: Переходим в каталог и создаем там файл lab7-1.asm

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.1(рис. 3.2).



```
mc [vvustinova@rudn]:~/work/arch-pc/lab07
lab7-1.asm  [-M--]  4 L:[ 1+12 13/ 21] *(324 / 449b) 0095 0x05F
%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.
call sprintf.
_label2:
mov eax, msg2.
call sprintf.
_label3:
mov eax, msg3.
call sprintf.
_end:
call quit.
```

Рис. 3.2: Заполняем данный файл

Запускаем файл(рис. 3.3).

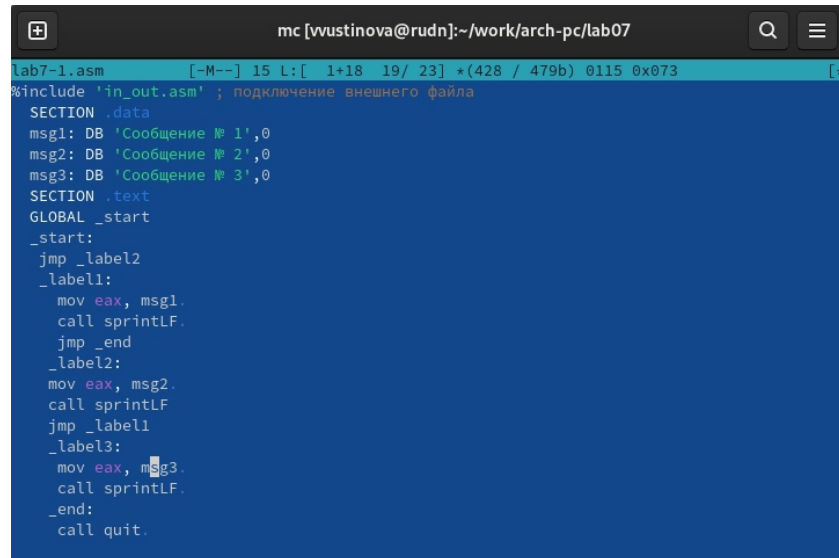
```

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

```

Рис. 3.3: Смотрим как работает файл

Снова открываем файл для редактирования и изменяем его в соответствии с листингом 7.2(рис. 3.4).



```

lab7-1.asm  [-M--] 15 L: [ 1+18 19/ 23] *(428 / 479b) 0115 0x073  [+
#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.
call sprintLF.
jmp _end
_label2:
mov eax, msg2.
call sprintLF
jmp _label1
_label3:
mov eax, msg3.
call sprintLF.
_end:
call quit.

```

Рис. 3.4: Редактируем файл

Запускаем файл(рис. 3.5).

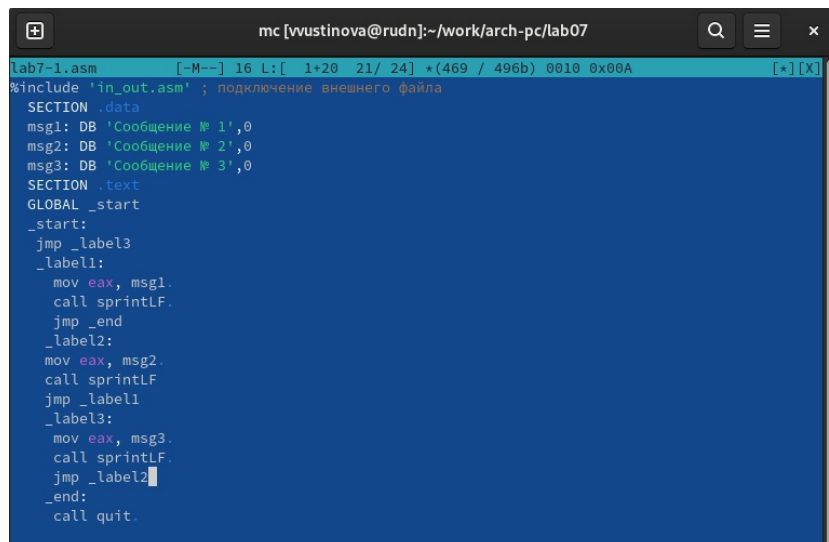
```

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

```

Рис. 3.5: Смотрим как работает файл

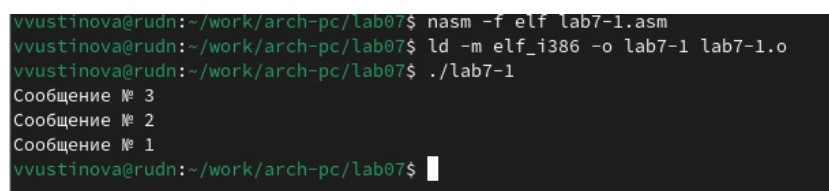
Требуется снова отредактировать файл(рис. 3.6).



```
lab7-1.asm [-M--] 16 L: [ 1+20 21/ 24] *(469 / 496b) 0010 0x00A [*] [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 _label3
_label1:
mov eax, msg1.
call sprintf.
jmp _end
_label2:
mov eax, msg2.
call sprintf.
jmp _label1
_label3:
mov eax, msg3.
call sprintf.
jmp _label2
_end:
call quit.
```

Рис. 3.6: Редактируем данный файл

Запускаем файл(рис. 3.7).



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

Рис. 3.7: Сверяемся с нужным выводом, все верно

Создаем файл Lab7-2.asm(рис. 3.8).



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

Рис. 3.8: Используем команду touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 7.3(рис. 3.9).

```

mc [vvustinova@rudn]:~/work/arch-pc/lab07
lab7-2.asm [----] 50 L: [ 6+14 20/ 50] *(432 /1345b) 0010 0x00A [*]
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].
jg check_B.
mov ecx,[C].
mov [max],ecx.
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi.
mov [max],eax.
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B].
jg fin.
mov ecx,[B].
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '

```

Рис. 3.9: Заполняем файл как указано в листинге

Запускаем файл и вводим различные значения(рис. 3.10).

```

vvustinova@rudn:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vvustinova@rudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vvustinova@rudn:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 15
Наибольшее число: 50
vvustinova@rudn:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
vvustinova@rudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
vvustinova@rudn:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 9
Наибольшее число: 50
vvustinova@rudn:~/work/arch-pc/lab07$

```

Рис. 3.10: Смотрим как работает наш файл

Изучение структуры файла листинга

Создаем файл листинга и открываем его в редакторе(рис. 3.11).

```

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

```

Рис. 3.11: Используем Mcedit

```

lab7-2.lst [----] 0 L: [ 1+ 0 1/225] *(0 /14059b) 0032 0x020 [*][X]
1                                     %include 'in_out.asm'
1                                     <1> ;----- slen -----
2                                     <1> ; Функция вычисления длины сообщения
3                                     <1> slen:.....
4 00000000 53                         <1>      push    ebx.....
5 00000001 89C3                       <1>      mov     ebx, eax.....
6                                     <1> .....
7                                     <1> nextchar:.....
8 00000003 803800                     <1>      cmp     byte [eax], 0...
9 00000006 7403                       <1>      jz      finished.....
10 00000008 40                        <1>      inc     eax.....
11 00000009 EBF8                      <1>      jmp     nextchar.....
12                                     <1> .....
13                                     <1> finished:
14 0000000B 29D8                     <1>      sub     eax, ebx
15 0000000D 5B                        <1>      pop     ebx.....
16 0000000E C3                       <1>      ret.....
17                                     <1> .
18                                     <1> .
19                                     <1> ;----- sprint -----
20                                     <1> ; Функция печати сообщения
21                                     <1> ; входные данные: mov eax,<message>
22                                     <1> sprint:
23 0000000F 52                         <1>      push    edx
24 00000010 51                         <1>      push    ecx
25 00000011 53                         <1>      push    ebx
26 00000012 50                         <1>      push    eax
27 00000013 E8E8FFFFFF                <1>      call    slen
28                                     <1> .....
29 00000018 89C2                     <1>      mov     edx, eax
30 0000001A 58                        <1>      pop     eax
31                                     <1> .....
32 0000001B 89C1                     <1>      mov     ecx, eax
33 0000001D BB01000000                <1>      mov     ebx, 1

```

Рис. 3.12: Отрываем файл

1. Строка 14: 0000000B 29D8 sub eax, ebx Описание: 0000000B — адрес команды в сегменте кода (в памяти). 29D8 — машинный код инструкции. Это бинарное представление команды sub eax, ebx. sub eax, ebx — команда процессора, которая вычитает значение регистра ebx из регистра eax и записывает результат обратно в eax.
2. 2 Строка 28: E8E8FFFFFF call slen Описание: E8E8FFFFFF — машинный код инструкции вызова функции. Команда call записывает текущий адрес (следующий за вызовом) в стек и передаёт управление указанной функции. call slen — вызов функции slen. Эта функция (определена выше в коде) вычисляет длину строки, переданной в eax.

3. 3 Строка 33: 0000001D BB01000000 mov ebx, 1 Описание:0000001D — адрес команды в сегменте кода.BB01000000 — машинный код инструкции, который соответствует команде mov ebx, 1.mov ebx, 1 — команда, которая загружает значение 1 в регистр ebx.

Открываем файл и удаляем один операндум(рис. 3.13).

```

lab7-2.asm      [-M--]  7 L:[ 1+17 18/ 50] *(338
#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,
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].
jg check_B.
mov ecx,[C].
mov [max],ecx.
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi.
mov [max],eax.
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B].
jg fin.
mov ecx,[B].

```

Рис. 3.13: Удалили операндум

```

vvustinova@rudn:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:18: error: invalid combination of opcode and operands
vvustinova@rudn:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
vvustinova@rudn:~/work/arch-pc/lab07$

```

Рис. 3.14: Транслируем файл

В файле листинга показывает ошибку, при запуске. Никакие входные файлы помимо файла листинга не создаются (рис. 3.15).

```

lab7-2.lst [B---] 90 L:[181+13 194/226] *(11903/14147b) 0010 0x00A [X]
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res Ah> max resb 10
9 0000000A <res Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 000000E8 B8[00000000] mov eax,msg1
15 000000ED E81DFFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 E847FFFFFF call sread
19 ; ----- Преобразование 'B' из символа в число
20 000000FC B8[0A000000] mov eax,B
21 00000101 E896FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
22 00000106 A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
23 ; ----- Записываем 'A' в переменную 'max'
24 00000108 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
25 00000111 890D[00000000] mov [max],ecx ; 'max = A'
26 ; ----- Сравниваем 'A' и 'C' (как символы)
27 00000117 3B0D[39000000] cmp ecx,[C]
28 0000011D 7F0C jg check_B
29 0000011F 8B0D[39000000] mov ecx,[C]

```

Рис. 3.15: Просматриваем ошибку в файле листинга

Задания для самостоятельной работы

Вариант 12

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, x и . Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу.

```

vvustinova@rudn:~/work/arch-pc/lab07$ touch lab7-3.asm
vvustinova@rudn:~/work/arch-pc/lab07$ mc

```

Рис. 3.16: Создаем файл

Открываем его и пишем программу, которая выберет наименьшее число из трех (рис. 3.17).

```

lab7-3.asm      [----]  2 L: [ 1+ 0  1/ 46] *(2  /1217b) 0110 0x06E      [*][X]
%include 'in_out.asm'
section .data
msg1 db 'Введите B: ',0h
msg2 db "Наименьшее число: ",0h
A dd '99'
C dd '29'
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] ; 'есх = A'
mov [min],ecx ; 'тах = A'
; ----- Сравниваем 'A'
cmp ecx,[C] ; Сравниваем 'A' и 'C'
jl check_B ; если 'A>C', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'есх = C'
mov [min],ecx ; 'тах = C'
; ----- Преобразование 'тах(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 'тах'
; ----- Сравниваем 'тах(A,C)' и 'B' (как числа)
mov ecx,[min]
cmp ecx,[B]
jl fin ; если 'тах(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'есх = B'
mov [min],ecx
; ----- Вывод результата
fin:
mov eax, msg2

```

Рис. 3.17: Сама программа

```

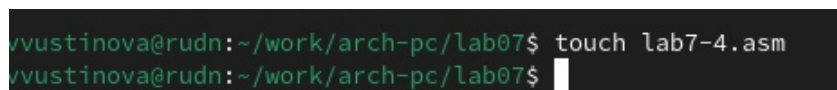
vvustinova@rudn:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
vvustinova@rudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
vvustinova@rudn:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 26
Наименьшее число: 26
vvustinova@rudn:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
vvustinova@rudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
vvustinova@rudn:~/work/arch-pc/lab07$ ./lab7-3
Введите B: 30
Наименьшее число: 29
vvustinova@rudn:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
vvustinova@rudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
vvustinova@rudn:~/work/arch-pc/lab07$ ./lab7-3

```

Рис. 3.18: Проверяме работу

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(x)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной ра-

боты № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6



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

Рис. 3.19: Создаем новый файл

Пишем программу, которая решит систему уравнений, при данных, введенных в консоль(рис. 3.20).


```

lab7-4.asm      [----]  7 L:[  1+ 2   3/ 61] *(30 / 962b) 0
#include 'in_out.asm'

SECTION .data
msg1: DB 'Введите x: ', 0h
msg2: DB 'Введите a: ', 0h
ans:  DB 'Результат системы: ', 0h

SECTION .bss
x: RESB 80
a: RESB 80
res: RESB 80

SECTION .text
GLOBAL _start
_start:
; Считываем значение x
mov eax, msg1
call sprint
mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi
mov [x], eax
; Считываем значение a
mov eax, msg2
call sprint
mov ecx, a
mov edx, 80
call sread

mov eax, a
call atoi
mov [a], eax

; Условие: если x < 5, то вычисляем a * x
mov eax, [x]
cmp eax, 5
jl less_than_5 ; Переход, если x < 5

```

Рис. 3.20: Пишем новую программу, для заданных нам условий

```
vvustinovalrudn:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
vvustinovalrudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
vvustinovalrudn:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 7
Результат системы: 21
vvustinovalrudn:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
vvustinovalrudn:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
vvustinovalrudn:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 6
Введите a: 4
Результат системы: 1
```

Рис. 3.21: Проверяем выполнение, все сделано корректно

4 Выводы

Мы познакомились с структурой файла листинга, изучили команды условного и безусловного перехода.