

# **Отчёт по лабораторной работе №7**

**Дисциплина: архитектура компьютера**

Чернятьева Олеся Олеговна

# Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
3	Сомнительная работа	12
4	Выводы	20

# Список иллюстраций

2.1	Создание директории . . . . .	5
2.2	Создание копии файла для дальнейшей работы . . . . .	5
2.3	Редактирование файла . . . . .	6
2.4	Запуск исполняемого файла . . . . .	6
2.5	Редактирование файла . . . . .	7
2.6	Запуск исполняемого файла . . . . .	7
2.7	Редактирование программы . . . . .	8
2.8	Создание исполняемого файла . . . . .	8
2.9	Создание файла . . . . .	8
2.10	Вставляю текст в файл . . . . .	9
2.11	Запуск исполняемого файла . . . . .	9
2.12	Редактирование файла . . . . .	10
2.13	Файл листинга . . . . .	10
2.14	Файл листинга . . . . .	11
2.15	Файл листинга . . . . .	11
3.1	Создание файла . . . . .	12
3.2	Редактирование файла . . . . .	13
3.3	Запуск исполняемого файла . . . . .	13
3.4	создание файла . . . . .	16
3.5	ввод программы в файл . . . . .	16
3.6	Создание исполняемого файла . . . . .	17
3.7	запуск исполняемого файла . . . . .	17

# 1 Цель работы

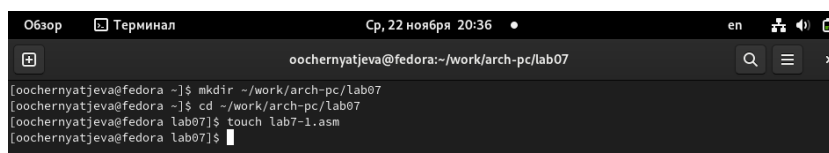
Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга

---

## 2 Выполнение лабораторной работы

1

С помощью утилиты `mkdir` создаю директорию `lab07`, перехожу в нее и создаю файл для работы. (рис. [2.1])



```
Обзор Терминал Ср, 22 ноября 20:36 en
oochernyatjeva@fedora:~/work/arch-pc/lab07
[oochernyatjeva@fedora ~]$ mkdir ~/work/arch-pc/lab07
[oochernyatjeva@fedora ~]$ cd ~/work/arch-pc/lab07
[oochernyatjeva@fedora lab07]$ touch lab7-1.asm
[oochernyatjeva@fedora lab07]$
```

Рис. 2.1: Создание директории

2

Копирую в текущий каталог файл `in_out.asm` из загрузок, т.к. он будет использоваться в других программах (рис. [2.2]).

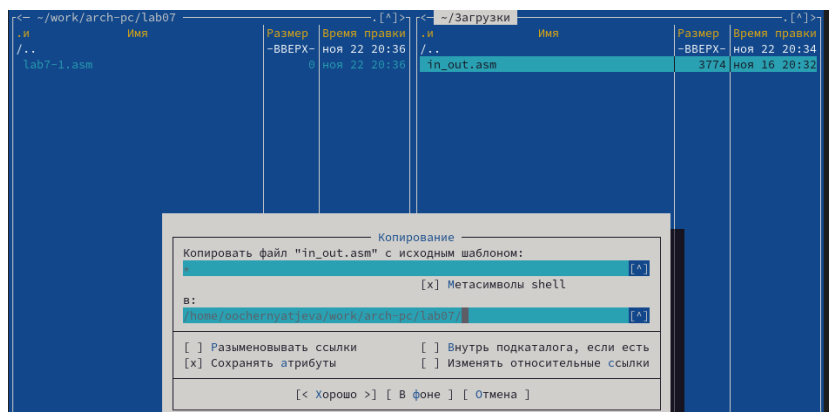
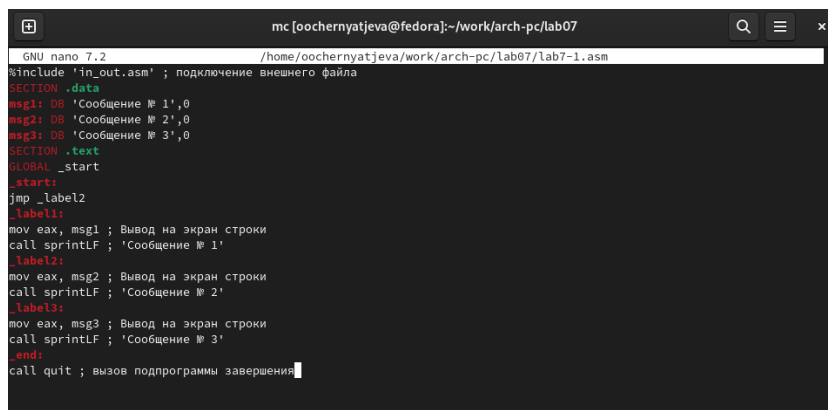


Рис. 2.2: Создание копии файла для дальнейшей работы

3

Открываю созданный файл lab7-1.asm, вставляю в него программу реализации безусловных переходов(рис. [2.3]).

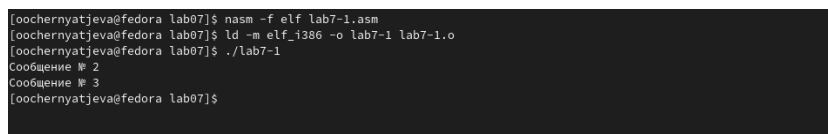


```
mc [oochernyatjeva@fedora]:~/work/arch-pc/lab07
GNU nano 7.2 /home/oochernyatjeva/work/arch-pc/lab07/lab7-1.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call sprintf ; 'Сообщение № 1'
_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintf ; 'Сообщение № 2'
_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintf ; 'Сообщение № 3'
end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.3: Редактирование файла

#### 4

Создаю исполняемый файл программы и запускаю его (рис. [2.4]). Инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`.

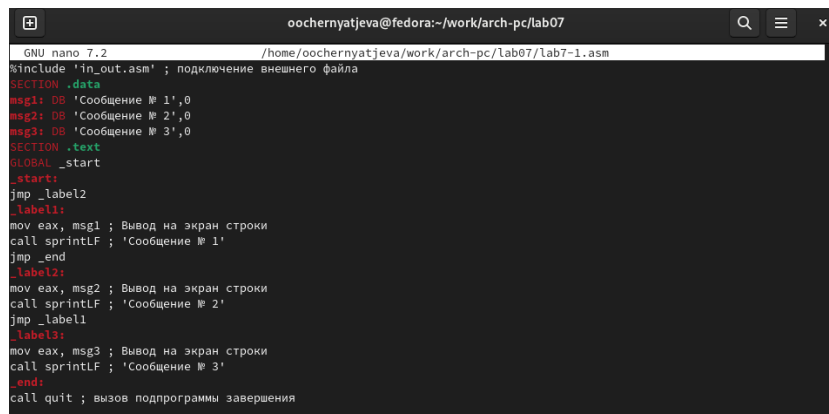


```
[oochernyatjeva@fedora lab07]$ nasm -f elf lab7-1.asm
[oochernyatjeva@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[oochernyatjeva@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 3
[oochernyatjeva@fedora lab07]$
```

Рис. 2.4: Запуск исполняемого файла

#### 5

Изменяю текст программы так, чтобы она выводила сначала 'Сообщение № 2', потом 'Сообщение № 1' и завершала работу (рис. [2.5]).

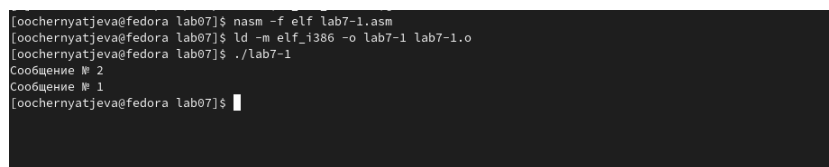


```
GNU nano 7.2 /home/oochernyatjeva/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label2
_label1:
mov eax, msg1 ; Вывод на экран строки
call printf ; 'Сообщение № 1'
jmp _end
_label2:
mov eax, msg2 ; Вывод на экран строки
call printf ; 'Сообщение № 2'
jmp _label1
_label3:
mov eax, msg3 ; Вывод на экран строки
call printf ; 'Сообщение № 3'
_end:
call quit ; вызов подпрограммы завершения
```

Рис. 2.5: Редактирование файла

## 6

Создаю новый исполняемый файл программы и запускаю его (рис. [2.6]). Убеждаюсь в том, программа работает верно.

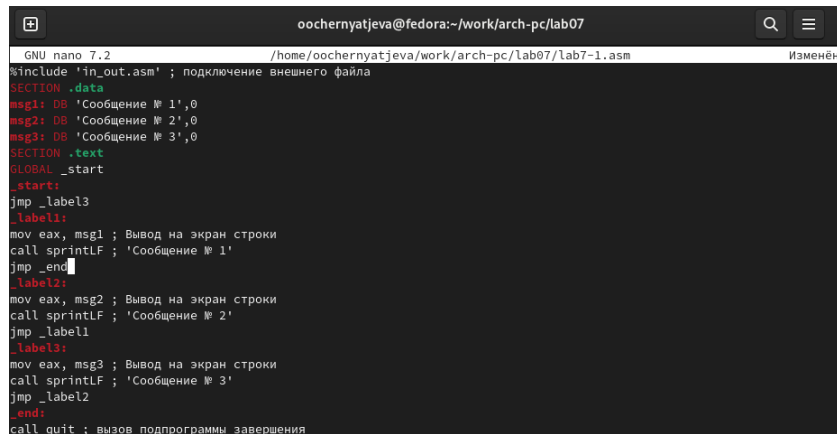


```
[oochernyatjeva@fedora lab07]$ nasm -f elf lab7-1.asm
[oochernyatjeva@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[oochernyatjeva@fedora lab07]$ ./lab7-1
Сообщение № 2
Сообщение № 1
[oochernyatjeva@fedora lab07]$
```

Рис. 2.6: Запуск исполняемого файла

## 7

Изменяю текст программы, так чтобы вывод происходил в обратном порядке (рис. [2.7]).

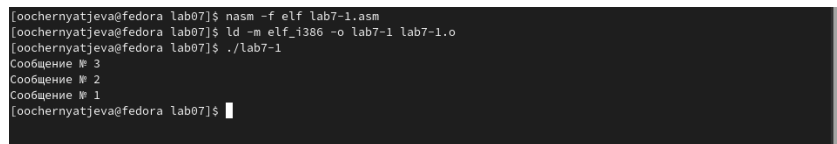


```
GNU nano 7.2 /home/oochernyatjeva/work/arch-pc/lab07/lab7-1.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start
_start:
jmp _label3
_label1:
mov eax, msg1 ; Вывод на экран 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. 2.7: Редактирование программы

## 8

Создаю исполняемый файл и проверяю работу программы (рис. [2.8]). Программа отработало верно.



```
[oochernyatjeva@fedora lab07]$ nasm -f elf lab7-1.asm
[oochernyatjeva@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[oochernyatjeva@fedora lab07]$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
[oochernyatjeva@fedora lab07]$
```

Рис. 2.8: Создание исполняемого файла

## 9

Создаю новый файл lab7-2.asm для программы с условным оператором. (рис. [2.9]).



```
[oochernyatjeva@fedora lab07]$ touch lab7-2.asm
[oochernyatjeva@fedora lab07]$ mc
```

Рис. 2.9: Создание файла

## 10

Вставляю программу, которая определяет и выводит на экран наибольшее число (рис.[2.10]).



```

GNU nano 7.2 /home/oochernyatjeva/work/arch-pc/lab07/lab7-2.asm
;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
; ----- Вывод результата

```

Рис. 2.10: Вставляю текст в файл

## 11

Создаю и запускаю новый исполняемый файл, проверяю работу программы для разных B, при A=20 и C=50 (рис. [2.11]).

```

[oochernyatjeva@fedora lab07]$ nasm -f elf lab7-2.asm
[oochernyatjeva@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[oochernyatjeva@fedora lab07]$ ./lab7-2
Введите B: 60
Наибольшее число: 60
[oochernyatjeva@fedora lab07]$ ./lab7-2
Введите B: 10
Наибольшее число: 10
[oochernyatjeva@fedora lab07]$ ./lab7-2
Введите B: 50
Наибольшее число: 50
[oochernyatjeva@fedora lab07]$ ./lab7-2
Введите B: 20
Наибольшее число: 20

```

Рис. 2.11: Запуск исполняемого файла

## 12

Создаю файл листинга для программы в файле lab7-2.asm (рис. [2.12]).

```
oochernyatjeva@fedora lab07]$ nasm -f elf -l lab7-2.lst lab7-2.asm
oochernyatjeva@fedora lab07]$ mcedit lab7-2.lst
```

Рис. 2.12: Редактирование файла

13

Открываю файл листинга... Рассмотрим 17 строку подробно: (рис. [2.13]).

```
lab7-2.lst [----] 0 L:[181+44 225/225] *(14458/14458b) <EOF> [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 E810FFFFFF call sprint
16 ; ----- Ввод 'B'
17 000000F2 B9[0A000000] mov ecx,B
18 000000F7 BA0A000000 mov edx,10
19 000000FC E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000] mov eax,B
22 00000106 E91FFFFFFF call atoi ; Вызов подпрограммы перевода символа в число
23 0000010B A3[0A000000] mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A] ; 'ecx = A'
26 00000116 890D[00000000] mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C jg check_B ; если 'A>C', то переход на метку 'check_B',
30 00000124 8B0D[39000000] mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000] mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF call atoi ; Вызов подпрограммы перевода символа в число
36 0000013A A3[00000000] mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
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 на 'fin',
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 E8ACFFFFFF call sprint ; Вывод сообщения 'Наибольшее число: '
47 00000163 A1[00000000] mov eax,[max]
48 00000168 E819FFFFFF call iprintLF ; Вывод 'max(A,B,C)'
49 0000016D E869FFFFFF call quit ; Выход
```

Рис. 2.13: Файл листинга

17 строка:

- Первые цифры [17] - это номер строки файла листинга.
- Следующие цифры [0000000F2] адрес — это смещение машинного кода от начала текущего сегмента, состоит из 8 чисел.
- следующие числа {B9[0A000000]} - это машинный код, который представляет собой ассемблированную исходную строку [mov ecx,b] в виде шестна-

дцатеричной последовательности, поэтому и появляются буквы латинского алфавита.

- следующее [mov ecx,b] - исходный текст программы, которая просто состоит из строк исходной программы вместе с комментариями.

## 14

Открываю файл lab7-2.asm с помощью редактора и Удаляю один операнд в инструкции cmp. (рис. [2.14]).

```
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx, ; 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
; ----- Вывод результата
fin:
```

Рис. 2.14: Файл листинга

## 15

Открываю файл листинга с помощью редактора mscedit и замечаю, что в файле листинга появляется ошибка. (рис. [2.15]).

```
38 0000013F 8B0D[00000000]      mov ecx,[max]
39                               cmp ecx, ; Сравниваем 'max(A,C)' и 'B'
39                               error: invalid combination of opcode and operands
40 00000145 7F0C                      jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 00000147 8B0D[0A000000]      mov ecx,[B] ; иначе 'ecx = B'
42 0000014D 890D[00000000]      mov [max],ecx
43                               ; ----- Вывод результата
44                               fin:
45 00000153 B8[13000000]      mov eax, msg2
46 00000158 E8B2FEFFFF      call sprint ; Вывод сообщения 'Наибольшее число: '
47 0000015D A1[00000000]      mov eax,[max]
48 00000162 E81FFFEFFF      call iprintLF ; Вывод 'max(A,B,C)'
49 00000167 E86FFFEFFF      call quit ; Выход
```


Рис. 2.15: Файл листинга

Отсюда можно сделать вывод, что, если в коде появляется ошибка, то ее описание появится в файле листинга

## 3 Сомтоятельная работа

1

Создаю файл lab7-3.asm с помощью утилиты touch (рис. [3.1]).

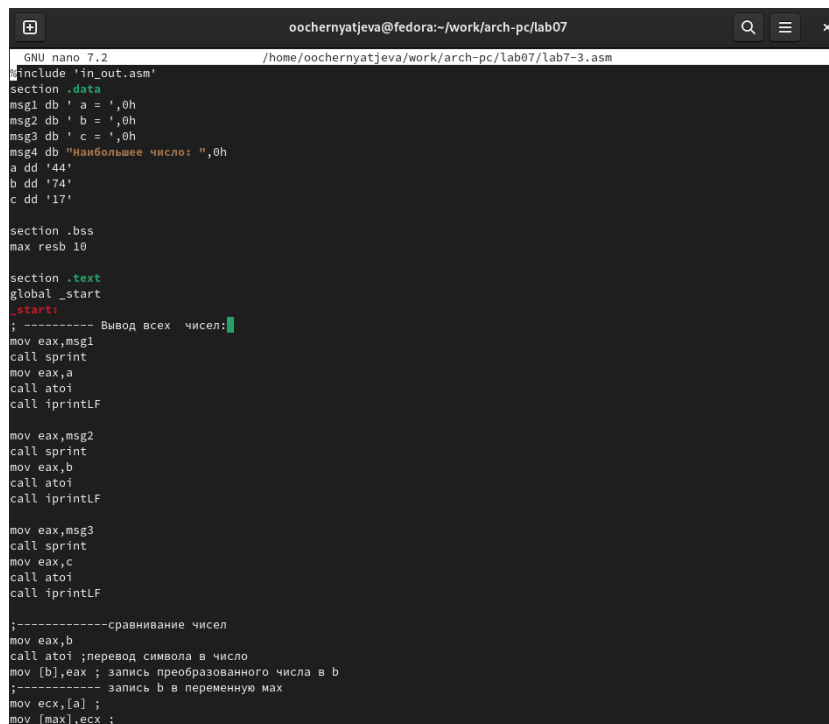


```
oochernyatjeva@fedora lab07]$ touch lab7-3.asm  
oochernyatjeva@fedora lab07]$ mcedit lab7-3.asm
```

Рис. 3.1: Создание файла

2

Ввожу в созданный файл текст программы для вычисления наибольшего из 3 чисел. Числа беру, учитывая свой вариант из прошлой лабораторной работы. 2 вариант (рис. [3.2]).



```
GNU nano 7.2 /home/oochernyatjeva/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
msg4 db "Наибольшее число: ",0h
a dd '44'
b dd '74'
c dd '17'

section .bss
max resb 10

section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF

mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF

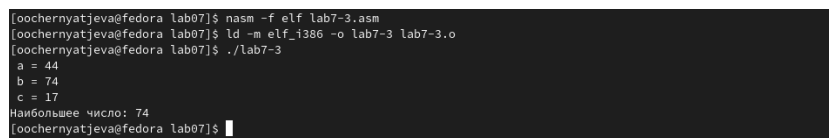
mov eax,msg3
call sprint
mov eax,c
call atoi
call iprintLF

;-----сравнение чисел
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov ecx,[a]
mov [max],ecx ;
```

Рис. 3.2: Редактирование файла

### 3

Создаю исполняемый файл и запускаю его (рис. [3.3]).



```
[oochernyatjeva@fedora lab07]$ nasm -f elf lab7-3.asm
[oochernyatjeva@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[oochernyatjeva@fedora lab07]$ ./lab7-3
a = 44
b = 74
c = 17
Наибольшее число: 74
[oochernyatjeva@fedora lab07]$
```

Рис. 3.3: Запуск исполняемого файла

### Текст программы

```
%include 'in_out.asm'

section .data

msg1 db ' a = ',0h
msg2 db ' b = ',0h
msg3 db ' c = ',0h
```

```
msg4 db "Наибольшее число: ",0h
a dd '44'
b dd '74'
c dd '17'
```

```
section .bss
max resb 10
```

```
section .text
global _start
_start:
; ----- Вывод всех чисел:
mov eax,msg1
call sprint
mov eax,a
call atoi
call iprintLF

mov eax,msg2
call sprint
mov eax,b
call atoi
call iprintLF

mov eax,msg3
call sprint
mov eax,c
call atoi
call iprintLF
```

```

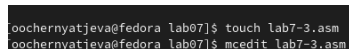
;-----сравнивание чисел
mov eax,b
call atoi ;перевод символа в число
mov [b],eax ; запись преобразованного числа в b
;----- запись b в переменную max
mov ecx,[a] ;
mov [max],ecx ;
;-----сравнивание чисел a c
cmp ecx,[c]; if a>c
jg check_b ; то перход на метку
mov ecx,[c] ;
mov [max],ecx ;
;-----метка check_b
check_b:
mov eax,max ;
call atoi
mov [max],eax ;
;-----
mov ecx,[max] ;
cmp ecx,[b] ;
jg check_c ;
mov ecx,[b] ;
mov [max],ecx ;
;-----
check_c:
mov eax,msg4 ;
call sprint ;
mov eax,[max];

```

```
call iprintLF ;  
call quit
```

4

Создаю новый файл lab7-4 для написания программы второго задания. (рис. [3.4]).

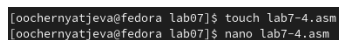


```
[oochernyatjeva@fedora lab07]$ touch lab7-3.asm  
[oochernyatjeva@fedora lab07]$ mcedit lab7-3.asm
```

Рис. 3.4: создание файла

5

Ввожу в него программу, (рис. [3.5]). в которую ввожу 2 значения x и a, и которая выводит значения функции. Функцию беру из таблицы в соответствии со своим вариантом (Вариант 16).



```
[oochernyatjeva@fedora lab07]$ touch lab7-4.asm  
[oochernyatjeva@fedora lab07]$ nano lab7-4.asm
```

Рис. 3.5: ввод программы в файл

6

Создаю исполняемый файл и проверяю её выполнение при x=5, a=7 (рис. [3.6]). Программа отработала верно!



```

oochernyatjeva@fedora:~/work/arch-pc/lab07 — nano lab7-4.asm
GNU nano 7.2 lab7-4.asm

section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,10
call sread
mov eax,x
;-----
call atoi
mov [x],eax
;-----

mov eax,msg2
call sprint
mov ecx,a
mov edx,10
call sread
mov eax,a ;
call atoi
mov [a],eax ;
;-----
mov ecx,4
cmp ecx,[x] ; 4>x, то переход в check_a
jg check_a ;
mov eax,[x] ; иначе
mov ebx,[a];
mul ebx ;
mov ecx,eax ;
jmp _end ;
check_a:
mov ecx,[x]
add ecx,4;
;-----
_end:
mov eax,msg3 ;
call sprint ;
mov eax,ecx ;
call iprintf;
call quit ;

```

Рис. 3.6: Создание исполняемого файла

7

Повторный раз запускаю программу и проверяю ее выполнение при  $x=6$  и  $a=4$  (рис. [3.7]). Программа отработала верно!

```

oochernyatjeva@fedora:~/work/arch-pc/lab07
[oochernyatjeva@fedora lab07]$ nasm -f elf lab7-4.asm
[oochernyatjeva@fedora lab07]$ ld -m elf_i386 -o lab7-4 lab7-4.o
[oochernyatjeva@fedora lab07]$ ./lab7-4
Введите x: 1
Введите a: 1
f(x) = 5
[oochernyatjeva@fedora lab07]$ ./lab7-4
Введите x: 7
Введите a: 1
f(x) = 7
[oochernyatjeva@fedora lab07]$

```

Рис. 3.7: запуск исполняемого файла

## Текст программы

```

#include 'in_out.asm'

section .data

```

```
msg1 db 'Введите x: ',0h
msg2 db 'Введите a: ',0h
msg3 db 'f(x) = ',0h
```

```
section .bss
```

```
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
```

```
mov [x],eax
```

```
;-----
```

```
mov eax,msg2
```

```
call sprint
```

```
mov ecx,a
```

```
mov edx,10
```

```
call sread
```

```
mov eax,a ;
```

```
call atoi
```

```

mov [a],eax ;
;-----
mov ecx,4
cmp ecx,[x] ; 4>x, то переход в check_a
jg check_a ;
mov eax,[x] ; иначе
mov ebx,[a];
mul ebx ;
mov ecx,eax ;
jmp _end ;
check_a:
mov ecx,[x]
add ecx,4;
;-----
_end:
mov eax,msg3 ;
call sprint ;
mov eax,ecx ;
call iprintLF;
call quit ;

```

## 4 Выводы

При выполнении данной лабораторной работы я освоила инструкции условного и безусловного вывода и ознакомился с структурой файла листинга.