

Отчёт по лабораторной работе №7

Дисциплина: Архитектура компьютера

Бызова Мария Олеговна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файлы листинга	11
3	Выполнение заданий для самостоятельной работы	14
4	Выводы	23

Список иллюстраций

2.1	Создание необходимой директории и файла	6
2.2	Редактирование файла	7
2.3	Создание исполняемого файла	7
2.4	Запуск исполняемого файла	7
2.5	Редактирование файла	8
2.6	Создание исполняемого файла	8
2.7	Запуск исполняемого файла	8
2.8	Редактирование файла	9
2.9	Создание исполняемого файла	9
2.10	Запуск исполняемого файла	9
2.11	Создание файла	10
2.12	Редактирование файла	10
2.13	Создание исполняемого файла	10
2.14	Запуск исполняемого файла	11
2.15	Запуск исполняемого файла	11
2.16	Создание файла листинга	11
2.17	Открытие файла листинга	11
2.18	Открытый файл листинга	12
2.19	Выбранные строки	12
2.20	Изменение файла	13
2.21	Создание файла листинга	13
2.22	Созданные файлы	13
2.23	Открытый файл листинга	13
3.1	Создание файла	14
3.2	Написание программы	15
3.3	Создание исполняемого файла	15
3.4	Запуск исполняемого файла	15
3.5	Создание файла	16
3.6	Написание программы	16
3.7	Создание исполняемого файла	16
3.8	Запуск исполняемого файла	16
3.9	Запуск исполняемого файла	17

Список таблиц

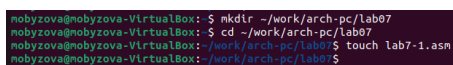
1 Цель работы

Целью лабораторной работы является изучение команд условного и безусловного переходов, приобретение навыков написания программ с использованием переходов, знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

1. С помощью утилиты `mkdir` создаем директорию, в которой будем создавать файлы с программами для лабораторной работы №7. Переходим в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаем файл `lab7-1.asm` (рис. [2.1]).



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

Рис. 2.1: Создание необходимой директории и файла

2. При помощи Midnight Commander открываем созданный файл `lab7-1.asm`, вставляем в него программу с использованием инструкции `jmp` из листинга 7.1 (рис. [2.2]).

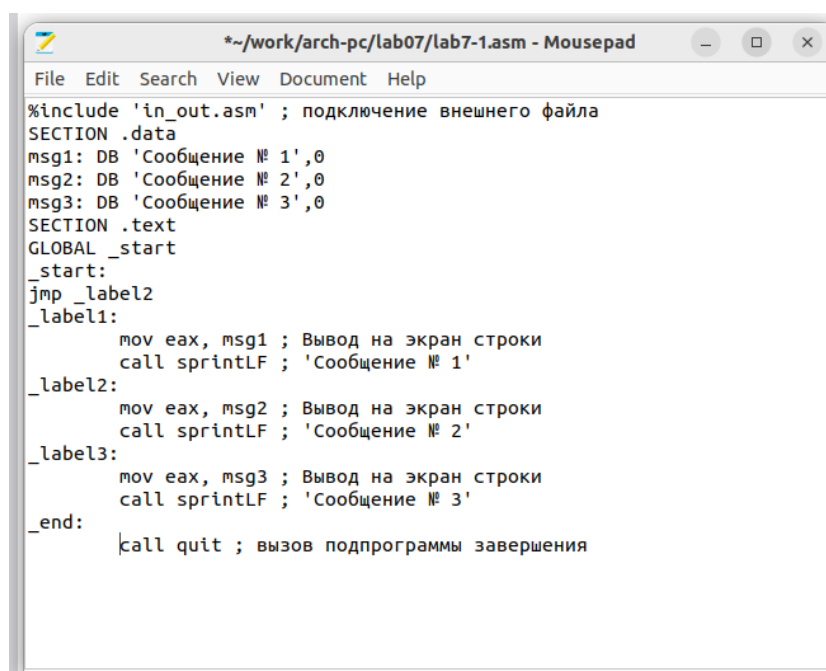


Рис. 2.2: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.3], [2.4]).

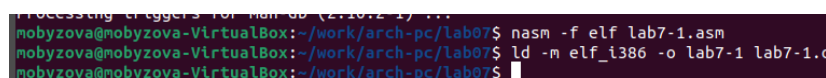


Рис. 2.3: Создание исполняемого файла

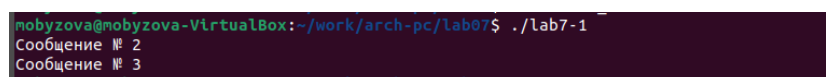
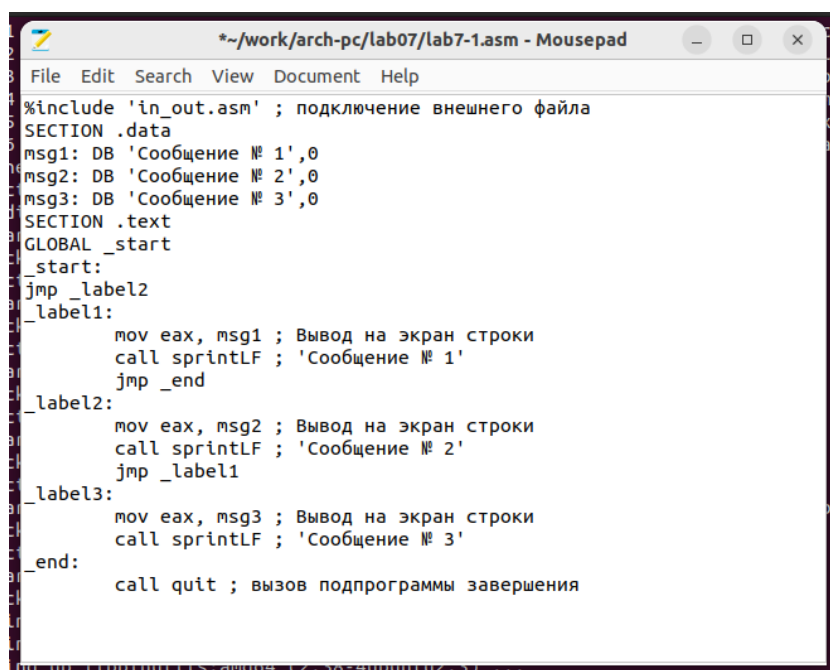


Рис. 2.4: Запуск исполняемого файла

Таким образом, использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Изменим текст программы в соответствии с листингом 7.2 таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим

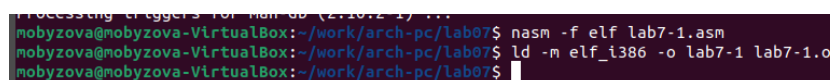
инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`) (рис. [2.5]).



```
*~/work/arch-pc/lab07/lab7-1.asm - Mousepad
File Edit Search View Document Help
%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
```

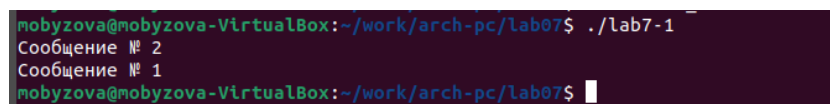
Рис. 2.5: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.6], [2.7]).



```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.6: Создание исполняемого файла

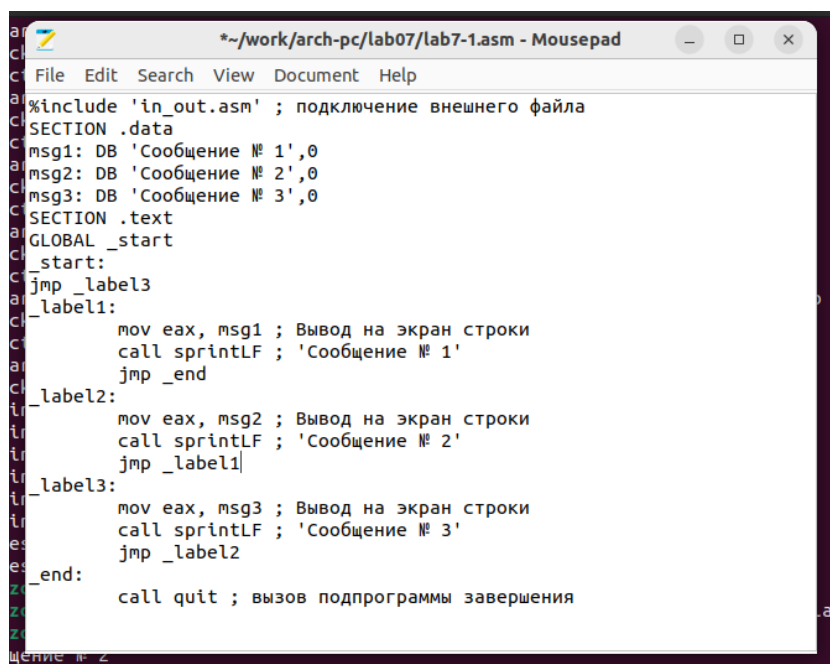


```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.7: Запуск исполняемого файла

Теперь изменим текст программы, добавив или изменив инструкции `jmp`, таким образом, чтобы она выводила сначала ‘Сообщение № 3’, потом ‘Сообщение

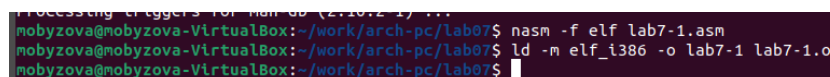
№ 2', потом 'Сообщение № 1' и завершала работу. (рис. [2.8]).



```
File Edit Search View Document Help
%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
```

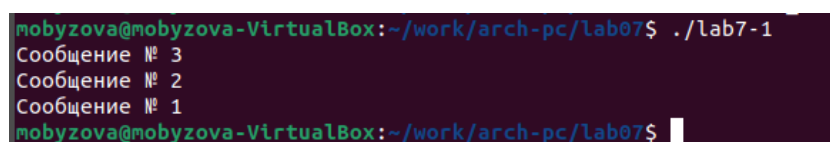
Рис. 2.8: Редактирование файла

Далее создаем исполняемый файл и запускаем его (рис. [2.9], [2.10]).



```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.9: Создание исполняемого файла



```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.10: Запуск исполняемого файла

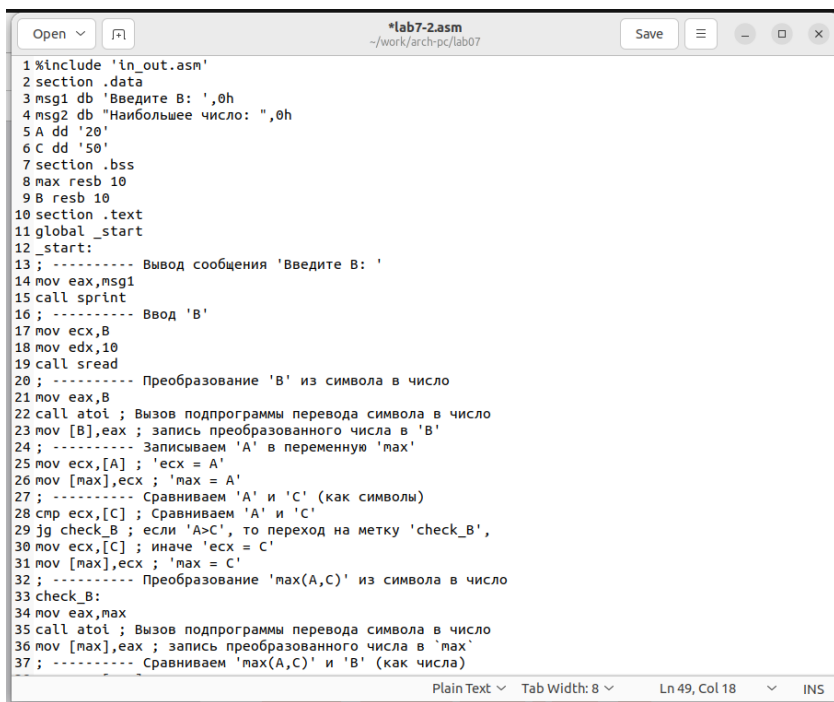
Убеждаемся, что программа отработала верно.

3. Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 (рис. [2.11]).

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

Рис. 2.11: Создание файла

Внимательно изучив текст программы из листинга 7.3, введем его в lab7-2.asm (рис. [2.12]).



```
*lab7-2.asm
~/work/arch-pc/lab07

1 %include 'in_out.asm'
2 section .data
3 msg1 db 'Введите B: ',0h
4 msg2 db "Наибольшее число: ",0h
5 A dd '20'
6 C dd '50'
7 section .bss
8 max resb 10
9 B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,msg1
15 call sprint
16 ; ----- Ввод 'B'
17 mov ecx,B
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,B
22 call atoi ; Вызов подпрограммы перевода символа в число
23 mov [B],eax ; запись преобразованного числа в 'B'
24 ; ----- Записываем 'A' в переменную 'max'
25 mov ecx,[A] ; 'ecx = A'
26 mov [max],ecx ; 'max = A'
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 jg check_B ; если 'A>C', то переход на метку 'check_B',
30 mov ecx,[C] ; иначе 'ecx = C'
31 mov [max],ecx ; 'max = C'
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 mov eax,max
35 call atoi ; Вызов подпрограммы перевода символа в число
36 mov [max],eax ; запись преобразованного числа в 'max'
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
```

Рис. 2.12: Редактирование файла

Далее создаем исполняемый файл и запускаем его. Проверим работу программы для разных значений B (рис. [2.13], [2.14], [2.15]).

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.13: Создание исполняемого файла

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 1
Наибольшее число: 50
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск исполняемого файла

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 100
Наибольшее число: 100
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск исполняемого файла

Убеждаемся, что программа отработала верно.

2.2 Изучение структуры файлы листинга

1. Создаем файл листинга для программы из файла lab7-2.asm, пользуясь следующей утилитой (рис. [2.16]).

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

Рис. 2.16: Создание файла листинга

Откроем файл листинга lab7-2.lst с помощью любого текстового редактора, например mcedit (рис. [2.17], [2.18]).

```
mobyzova@mobyzova-VirtualBox:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 2.17: Открытие файла листинга

```

1  #include "in_out.asm"
2
3  <1> ; функция вычисления длины сообщения
4  <1> slen:
5  00000000 53      <1> push    ebx
6  00000001 89C3     <1> mov     ebx, eax
7
8  <1> nextchar:
9  00000003 803800   <1> cmp     byte [eax], 0
10 00000006 7403     <1> jz      finished
11 00000008 40       <1> inc     eax
12 00000009 EBF8     <1> jmp     nextchar
13
14  <1> finished:
15 0000000B 29D8     <1> sub     eax, ebx
16 0000000D 5B       <1> pop     ebx
17 0000000E C3       <1> ret
18
19
20  <1> ; функция печати сообщения
21  <1> ; входные данные: mov eax, <message>
22
23  <1> sprintf:
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 E8E8FFFF <1> call    slen
29
30 00000018 89C2     <1> mov     edx, eax
31 0000001A 58       <1> pop     eax
32
33 0000001B 89C1     <1> mov     ecx, eax
34 0000001D 8B010000 <1> mov     ebx, 1
35 00000022 B8040000 <1> mov     eax, 4
36 00000027 CD80     <1> int     80h
37
38 00000029 5B       <1> pop     ebx

```

Рис. 2.18: Открытый файл листинга

Подробно опишем содержимое строк 17, 19, 21 (рис. [2.19]).

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	

Рис. 2.19: Выбранные строки

1. 17 (номер строки) 000000F2 (адрес, начинается по смещению 000000F2 в сегменте кода) B9[0A000000] (машинный код) move ecx,B (исходный текст программы, в котором мы помещаем значение, хранящееся в B, в ячейку ecx).
2. 19 (номер строки) 000000FC (адрес, начинается по смещению 000000FC в сегменте кода) E842FFFFFF (машинный код) call sread (исходный текст программы, в котором мы вызываем подпрограмму считывания введенного значения).
3. 21 (номер строки) 00000101 (адрес, начинается по смещению 00000101 в сегменте кода) B8[0A000000] (машинный код) mov eax,B (исходный текст программы, в котором мы помещаем значение, хранящееся в B, в ячейку eax).

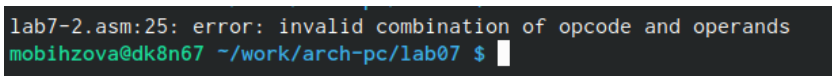
Затем зайдём в файл программы lab7-2.asm для удаления одного операнда. Удалим операнд в строчке: `mov ecx, [A]` (рис. [2.20]).



```
25      .....      mov ecx, ; 'ecx = A'
```

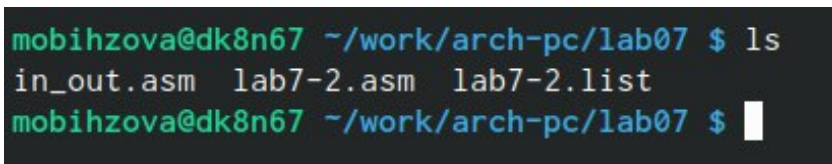
Рис. 2.20: Изменение файла

После сохранения изменений в программе проасsembлируем её с получением файла листинга. В ходе проасsembлирования система выдаёт ошибку и создаёт только файл lab7-2.list (рис. [2.21], [2.22]).



```
lab7-2.asm:25: error: invalid combination of opcode and operands
mobihzova@dk8n67 ~/work/arch-pc/lab07 $
```

Рис. 2.21: Создание файла листинга



```
mobihzova@dk8n67 ~/work/arch-pc/lab07 $ ls
in_out.asm  lab7-2.asm  lab7-2.list
mobihzova@dk8n67 ~/work/arch-pc/lab07 $
```

Рис. 2.22: Созданные файлы

Зайдем в листинг для изучения того, что добавилось в него, после возникновения ошибки (рис. [2.23]). Мы видим, что в него дополнительно добавляется строка “***** error: invalid combination of opcode and operands”, подтверждающая об ошибке.



```
mov ecx ; 'ecx = A'
error: invalid combination of opcode and operands
mov [max],ecx ; 'max = A'
```

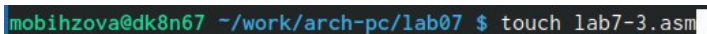
Рис. 2.23: Открытый файл листинга

3 Выполнение заданий для самостоятельной работы

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

Для написания программы возьмём за основу программу “Листинг 3”, в которую внесём изменения для правильного выполнения задания. В ходе написания программы будем использовать команду: JL (Переход если $a < b$).

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



```
mobihzova@dk8n67 ~/work/arch-pc/lab07 $ touch lab7-3.asm
```

Рис. 3.1: Создание файла

Открываем созданный файл для редактирования, вводим в него текст программы (рис. [3.3]).



```
File Edit Search View Document Help
~/work/arch-pc/lab07/lab7-3.asm - Mousepad

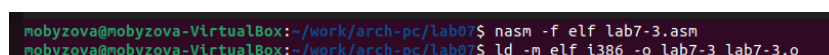
%include 'in_out.asm'
section .data
msg1 db 'Введите A: ',0h
msg2 db 'Введите B: ',0h
msg3 db 'Введите C: ',0h
msg4 db 'Наименьшее число: ',0h
section .bss
nln resb 10
A resb 10
B resb 10
C resb 10
section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,A
mov edx,10
-- -- --
```

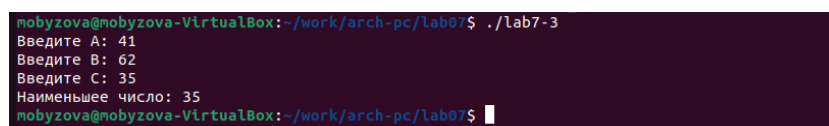
Рис. 3.2: Написание программы

Далее создаем исполняемый файл и запускаем его (рис. [3.4], [??]).



```
mobyzoa@mobyzoa-VirtualBox:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
mobyzoa@mobyzoa-VirtualBox:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
```

Рис. 3.3: Создание исполняемого файла



```
mobyzoa@mobyzoa-VirtualBox:~/work/arch-pc/lab07$ ./lab7-3
Введите A: 41
Введите B: 62
Введите C: 35
Наименьшее число: 35
mobyzoa@mobyzoa-VirtualBox:~/work/arch-pc/lab07$
```

Рис. 3.4: Запуск исполняемого файла

Введя числа из моего варианта №10, мы можем проверить правильность работы программы.

2. Задание: Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение заданной функции $f(a)$ и выводит результат вычислений. Вид функции $f(x)$ выбрать из таблицы 7.6 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу для значений x и a из 7.6.

Создаем файл lab7-3.asm с помощью утилиты touch (рис. [3.5]).

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

Рис. 3.5: Создание файла

Открываем созданный файл для редактирования, вводим в него текст программы вычисления функции: $x-2$ если $x>2$ и $3 \cdot a$ если $x \leq 2$ (выражение указано в соответствии с вариантом №10, полученным при выполнении лабораторной работы № 6).(рис. [3.6]).

```
~/work/study/arch-pc/lab07/lab7-4.asm - Mousepad
File Edit Search View Document Help
#include 'ln_out.asm'
SECTION .data
    msg1: DB 'Введите x: ',0h
    msg2: DB 'Введите a: ',0h
    otv: DB 'F(x) = ',0h
SECTION .bss
    x: RESB 80
    a: RESB 80
    res: RESB 80
    mIn: RESB 80
SECTION .text
GLOBAL _start
_start:

    mov eax,msg1
    call sprintf
    mov ecx,x
    mov edx,80
    call sread
    mov eax,x
    call atoi
    mov [x],eax

    mov eax,msg2
    call sprintf
    mov ecx,a
    mov edx,80
    call sread
    mov eax,a
    call atoi
    mov [a],eax

    mov ecx,[x]

    cmp ecx,2
    jle check_A
    jg check_X

check_A:
    ...
```

Рис. 3.6: Написание программы

Далее создаем исполняемый файл и запускаем его (рис. [3.7], [3.8], [3.9]).

```
mobyzova@mobyzova-VirtualBox: ~/work/study/arch-pc/lab07$ nasm -f elf lab7-4.asm
mobyzova@mobyzova-VirtualBox: ~/work/study/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
mobyzova@mobyzova-VirtualBox: ~/work/study/arch-pc/lab07$
```

Рис. 3.7: Создание исполняемого файла

```
mobyzova@mobyzova-VirtualBox: ~/work/study/arch-pc/lab07$ ./lab7-4
Введите x: 3
Введите a: 0
F(x) = 1
mobyzova@mobyzova-VirtualBox: ~/work/study/arch-pc/lab07$
```

Рис. 3.8: Запуск исполняемого файла


```
mobyzova@mobyzova-VirtualBox:~/work/study/arch-pc/lab0/$ ./lab7-4
Введите x: 1
Введите a: 2
F(x) = 6
mobyzova@mobyzova-VirtualBox:~/work/study/arch-pc/lab0/$
```

Рис. 3.9: Запуск исполняемого файла

Убеждаемся, что программа отработала верно.

Листинг №1. Изменённая программа 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 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'
    jmp _label2

_end:
    call quit ; вызов подпрограммы завершения
```

Листинг №2. Задание для самостоятельной работы №1

```
section .data
msg1 db 'Введите A: ',0h
msg2 db 'Введите B: ',0h
msg3 db 'Введите C: ',0h
msg4 db "Наименьшее число: ",0h

section .bss
min resb 10
A resb 10
B resb 10
C resb 10

section .text
global _start
_start:

mov eax,msg1
call sprint

mov ecx,A
mov edx,10
call sread

mov eax,A
call atoi
mov [A],eax

mov eax,msg2
call sprint
```

```
mov ecx,B
mov edx,10
call sread
```

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

```
mov eax,msg3
call sprint
```

```
mov ecx,C
mov edx,10
call sread
```

```
mov eax,C
call atoi
mov [C],eax
```

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

```
cmp ecx,[C]
jl check_B
mov ecx,[C]
mov [min],ecx
```

```
check_B:
```

```

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

fin:
    mov eax, msg4
    call sprint
    mov eax,[min]
    call iprintLF
    call quit

```

Листинг №3. Задание для самостоятельной работы №2

```

#include 'in_out.asm'

SECTION .data
    msg1: DB 'Введите x: ',0h
    msg2: DB 'Введите a: ',0h
    otv: DB 'F(x) = ',0h

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

SECTION .text
    GLOBAL _start
    _start:

    mov eax,msg1
    call sprint
    mov ecx,x

```

```

mov edx,80
call sread
mov eax,x
call atoi
mov [x],eax

mov eax,msg2
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
call atoi
mov [a],eax

mov ecx,[x]

cmp ecx,2
jle check_A
jg check_X

check_A:
mov eax,[a]
mov ebx, 3
mul ebx
mov [res],eax
jmp fin

check_X:

```

```
add ecx, -2
mov [res],ecx
jmp fin
```

```
fin:
mov eax,otv
call sprint
mov eax,[res]
call iprintLF
call quit
```

4 Выводы

В ходе выполнения лабораторной работы мы познакомились с назначением и структурой файла листинга, изучили команды условного и безусловного переходов, а также приобрели навыки написания программ с использованием переходов.