

# **Отчёт по лабораторной работе №7**

**Дисциплина: Архитектура компьютера**

Калашникова Ольга Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>7</b>
3.1	Реализация переходов в NASM . . . . .	7
3.2	Изучение структуры файлы листинга . . . . .	15
3.3	Задание для самостоятельной работы . . . . .	20
<b>4</b>	<b>Выводы</b>	<b>32</b>

## Список иллюстраций

3.1	Создание папки, её открытие и создание файла . . . . .	7
3.2	Проверка наличия папки . . . . .	7
3.3	Текст программы . . . . .	8
3.4	Перемещение файла “in_out.asm” . . . . .	8
3.5	Создание и запуск исполняемого файла . . . . .	9
3.6	Изменённый текст программы . . . . .	9
3.7	Изменённый текст программы . . . . .	10
3.8	Создание и запуск исполняемого файла . . . . .	10
3.9	Создание файла . . . . .	11
3.10	Проверка наличия файла . . . . .	12
3.11	Текст программы 1 . . . . .	13
3.12	Текст программы 1 . . . . .	14
3.13	Создание и запуск исполняемого файла . . . . .	14
3.14	Создание файла листинга . . . . .	15
3.15	Проверка наличия файла . . . . .	15
3.16	Открытие файла листинга 1 . . . . .	16
3.17	Открытие файла листинга 2 . . . . .	16
3.18	Строки листинга, которые будут описаны ниже . . . . .	16
3.19	Изменение кода . . . . .	17
3.20	Трансляция с получением файла листинга . . . . .	18
3.21	Созданный файл . . . . .	19
3.22	Файл листинга . . . . .	20
3.23	Изменение кода 1 . . . . .	21
3.24	Изменение кода 2 . . . . .	22
3.25	Создание и запуск исполняемого файла . . . . .	23
3.26	Значения в варианте . . . . .	23
3.27	Вариант 7 . . . . .	26
3.28	Текст программы 1 . . . . .	27
3.29	Текст программы 2 . . . . .	28
3.30	Создание и запуск исполняемого файла . . . . .	29

## 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

Здесь приводится описание задания в соответствии с рекомендациями методического пособия и выданным вариантом.

## 3 Выполнение лабораторной работы

### 3.1 Реализация переходов в NASM

Создаём каталог для программ лабораторной работы № 7 (при помощи команды `mkdir ~/work/arch-pc/lab07`), переходим в него (при помощи команды `cd ~/work/arch-pc/lab07`) и создаём файл `lab7-1.asm` (при помощи команды `touch lab7-1.asm`) (рис. 3.1),(рис. 3.2)

```
oskashnikova@dk6n62:~$ mkdir ~/work/arch-pc/lab07
oskashnikova@dk6n62:~$ cd ~/work/arch-pc/lab07
oskashnikova@dk6n62:~/work/arch-pc/lab07$ touch lab7-1.asm
oskashnikova@dk6n62:~/work/arch-pc/lab07$
```

Рис. 3.1: Создание папки, её открытие и создание файла

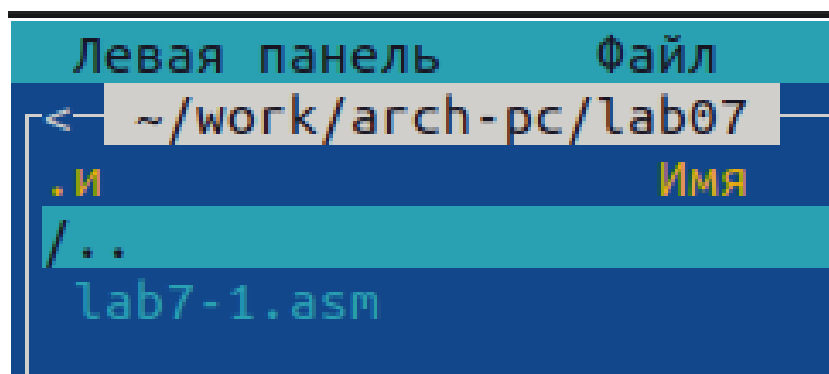


Рис. 3.2: Проверка наличия папки

Введём в файл `lab7-1.asm` текст программы из листинга 6.1., для корректной работы нужно переместить файл `in_out.asm` в тот же каталог, где лежит и файл

с текстом программы.(рис. 3.3),(рис. 3.4)

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

Рис. 3.3: Текст программы

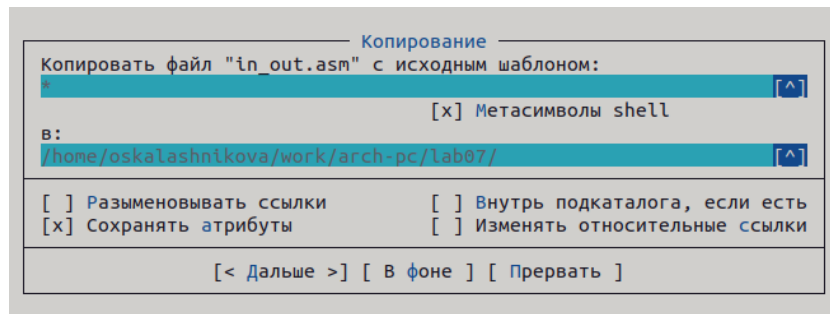


Рис. 3.4: Перемещение файла “in\_out.asm”

Создаем исполняемый файл и запускаем его (компиляция: `nasm -f elf lab7-1.asm`, `ld -m elf_i386 -o lab7-1 lab7-1.o`, запуск: `./lab7-1`) (рис. 3.5)



```
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ █
```

Рис. 3.5: Создание и запуск исполняемого файла

Мы видим, что программа вывела лишь 2 строки, потому что использование инструкции `jmp _label2` меняет порядок исполнения инструкций и позволяет выполнить инструкции начиная с метки `_label2`, пропустив вывод первого сообщения.

Меняем текст программы (чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу) (рис. 3.6)

```
/home/oskalashnikova/work/arch-pc/lab07/lab7-1-2.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'
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. 3.6: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab7-1-2.asm`, `ld -m elf_i386 -o lab7-1-2 lab7-1-2.o`, запуск: `./lab7-1-2`) (рис. ??)

Меняем текст программы (чтобы она выводила сначала 'Сообщение № 3', потом 'Сообщение № 2', затем 'Сообщение № 1' и завершала работу) (рис. 3.7)

```
/home/oskalashnikova/work/arch-pc/lab07/lab7-1-3.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 sprintfLF ; 'Сообщение № 1'
jmp _end

_label2:
mov eax, msg2 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 2'
jmp _label1

_label3:
mov eax, msg3 ; Вывод на экран строки
call sprintfLF ; 'Сообщение № 3'
jmp _label2

_end:
call quit ; вызов подпрограммы завершения
```

Рис. 3.7: Изменённый текст программы

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab7-1-3.asm`, `ld -m elf_i386 -o lab7-1-3 lab7-1-3.o`, запуск: `./lab7-1-3`) (рис. 3.8)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf lab7-1-3.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1-3 lab7-1-3.o
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-1-3
Сообщение № 3
Сообщение № 2
Сообщение № 1
oskalashnikova@dk6n62:~/work/arch-pc/lab07$
```

Рис. 3.8: Создание и запуск исполняемого файла

Для этого в начале используем инструкцию `jmp _label3`, меняя порядок испол-

нения инструкций и позволяя выполнить инструкции начиная с метки `_label3`. В текст программы после вывода сообщения № 3 добавим инструкцию `jmp` с меткой `_label2` (т.е. переход к инструкциям вывода сообщения № 2). После вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1). После вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Создаём файл “lab7-2.asm” в каталоге `~/work/arch-pc/lab07` (`touch ~/work/arch-pc/lab06/lab7-2.asm`) (рис. 3.9), (рис. 3.10)

```
oskashnikova@dk6n62:~/work/arch-pc/lab07$ touch ~/work/arch-pc/lab07/lab7-2.asm
oskashnikova@dk6n62:~/work/arch-pc/lab07$
```

Рис. 3.9: Создание файла

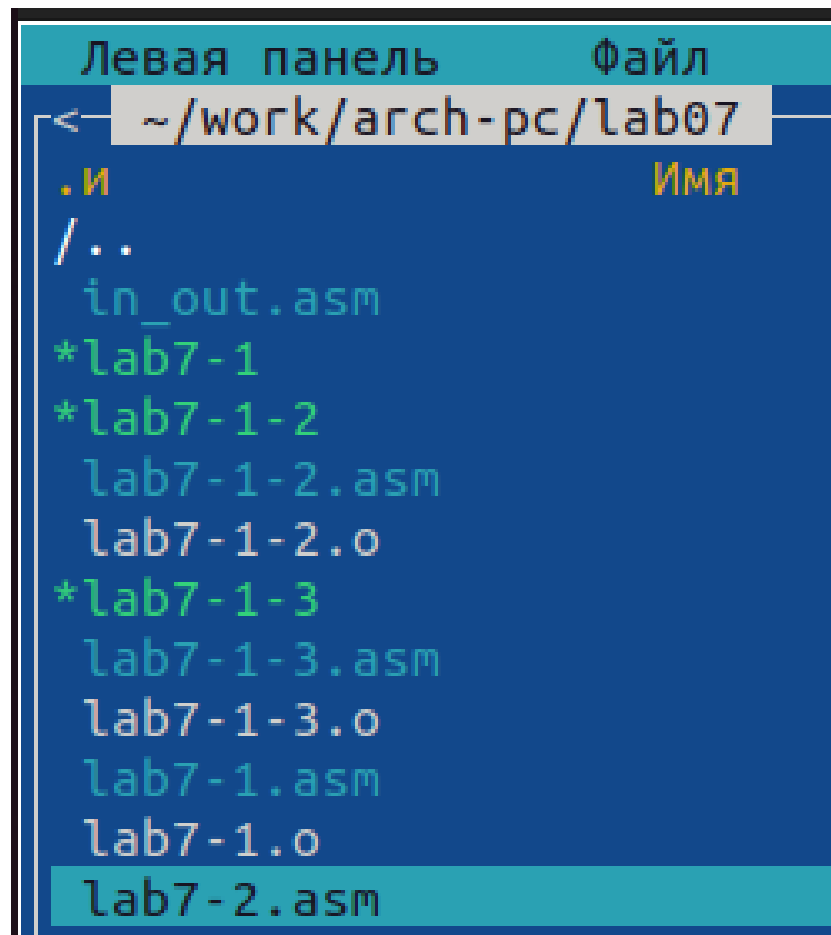


Рис. 3.10: Проверка наличия файла

Вводим в файл текст другой программы, 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) (рис. 3.11), (рис. 3.12)

```
/home/oskashnikova/work/arch-pc/lab07/lab7-2.asm [-M
#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:
```

Рис. 3.11: Текст программы 1

```

/home/oskalashnikova/work/arch-pc/lab07/lab7-2.asm [-M-]
; ----- 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
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

```

Рис. 3.12: Текст программы 1

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab7-2.asm`, `ld -m elf_i386 -o lab7-2 lab7-2.o`, запуск: `./lab7-2`) (рис. 3.13)

```

oskalashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 10
Наибольшее число: 50
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 51
Наибольшее число: 51
oskalashnikova@dk6n62:~/work/arch-pc/lab07$

```

Рис. 3.13: Создание и запуск исполняемого файла

## 3.2 Изучение структуры файлы листинга

Создадим файл листинга для программы из файла lab7-2.asm (nasm -f elf -l lab7-2.lst lab7-2.asm) (рис. 3.14),(рис. 3.15)

```
oskashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
oskashnikova@dk6n62:~/work/arch-pc/lab07$
```

Рис. 3.14: Создание файла листинга

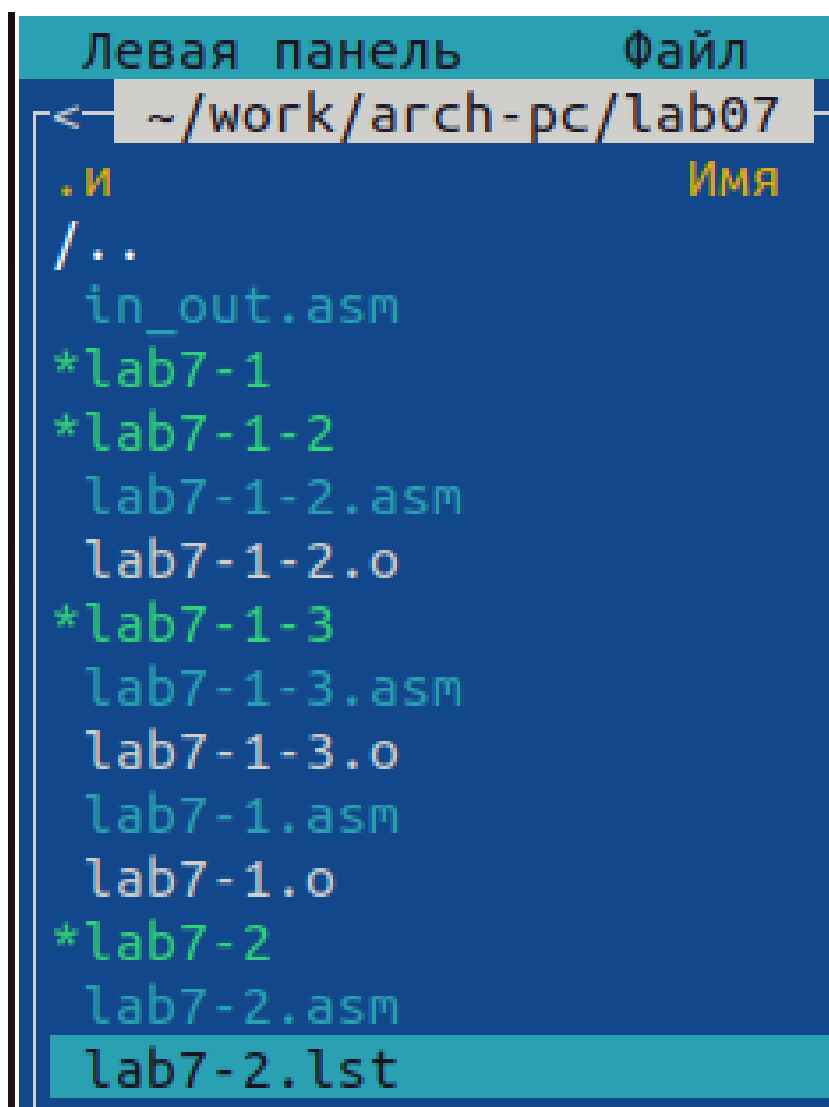


Рис. 3.15: Проверка наличия файла

Откроем файл листинга lab7-2.lst с помощью текстового редактора mcedit (mcedit lab7-2.lst) (рис. 3.16), (рис. 3.17)

```
|oskalashnikova@dk6n62:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 3.16: Открытие файла листинга 1

```
/home/oskalashnikova/work/arch-pc/lab07/lab7-2.lst [----] 27 L: [191+31 222/228] *(14198/14587b) 0032 0x020
16                                     ; ----- Вывод сообщения 'Введите B: '
17 000000E8 B8[00000000]             mov eax,msg1
18 000000ED E810FFFFFF               call sprint
19                                     ; ----- Ввод 'B'
20 000000F2 B9[0A000000]             mov ecx,B
21 000000F7 BA0A000000               mov edx,10
22 000000FC E842FFFFFF               call sread
23                                     ; ----- Преобразование 'B' из символа в число
24 00000101 B8[0A000000]             mov eax,B
25 00000106 E891FFFFFF               call atoi ; Вызов подпрограммы перевода символа в число
26 0000010B A3[0A000000]             mov [B],eax ; запись преобразованного числа в 'B'
27                                     ; ----- Записываем 'A' в переменную 'max'
28 00000110 8B0D[35000000]            mov ecx,[A] ; 'ecx = A'
29 00000116 890D[00000000]            mov [max],ecx ; 'max = A'
30                                     ; ----- Сравниваем 'A' и 'C' (как символы)
31 0000011C 3B0D[39000000]            cmp ecx,[C] ; Сравниваем 'A' и 'C'
32 00000122 7F0C                     jg check_B ; если 'A>C', то переход на метку 'check_B',
33 00000124 8B0D[39000000]            mov ecx,[C] ; иначе 'ecx = C'
34 0000012A 890D[00000000]            mov [max],ecx ; 'max = C'
35                                     ; ----- Преобразование 'max(A,C)' из символа в число
36                                     check_B:
37 00000130 B8[00000000]             mov eax,max
38 00000135 E862FFFFFF               call atoi ; Вызов подпрограммы перевода символа в число
39 0000013A A3[00000000]             mov [max],eax ; запись преобразованного числа в 'max'
40                                     ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
41 0000013F 8B0D[00000000]            mov ecx,[max]
42 00000145 3B0D[0A000000]            cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
43 0000014B 7F0C                     jg fln ; если 'max(A,C)>B', то переход на 'fln',
44 0000014D 8B0D[0A000000]            mov ecx,[B] ; иначе 'ecx = B'
45 00000153 890D[00000000]            mov [max],ecx
46                                     ; ----- Вывод результата
47                                     fln:
48 00000159 B8[13000000]             mov eax,msg2
49 0000015E E8ACFFFFFF               call sprint ; Вывод сообщения 'Наибольшее число: '
50 00000163 A1[00000000]             mov eax,[max]
51 00000168 E819FFFFFF               call fprintf ; Вывод 'max(A,B,C)'
52 0000016D E869FFFFFF               call quit ; Выход
```

Рис. 3.17: Открытие файла листинга 2

Опишем содержание строк 17, 21, 24 (рис. 3.18)

```
17 000000E8 B8[00000000]             mov eax,msg1
18 000000ED E810FFFFFF               call sprint
19                                     ; ----- Ввод 'B'
20 000000F2 B9[0A000000]             mov ecx,B
21 000000F7 BA0A000000               mov edx,10
22 000000FC E842FFFFFF               call sread
23                                     ; ----- Преобразование 'B' из символа в число
24 00000101 B8[0A000000]             mov eax,B
```

Рис. 3.18: Строки листинга, которые будут описаны ниже

17 000000E8 B8[00000000] mov eax,msg1

- 17: Это номер строки в исходном коде программы.



- 000000E8: Это адрес, по которому располагается данная инструкция в сегменте кода.
- B8[00000000]: Это машинный код операции mov для записи адреса msg1 в регистр eax.
- mov eax, msg1: Это исходное описание операции - она копирует значение, хранящееся по адресу msg1, в регистр eax.

21 000000F7 BA0A000000 mov edx,10

- 21: Номер строки.
- 000000F7: Адрес, где находится инструкция mov в сегменте кода.
- BA0A000000: Машинный код для операции mov, записывающей значение 10 в регистр edx.
- mov edx, 10: Исходный код программы, в котором мы помещаем значение 10 в регистр edx.

24 00000101 B8[0A000000] mov eax,B

- 24: Номер строки.
- 00000101: Адрес, где находится инструкция mov в сегменте кода.
- B8[0A000000]: Машинный код для операции mov, записывающей значение, хранящееся по адресу 0A000000 в регистр eax.
- mov eax, B: Исходный код программы, в котором мы помещаем значение, хранящееся по адресу 0A000000, в регистр eax.

Откроем файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалим один операнд. (рис. 3.19)



Рис. 3.19: Изменение кода

Выполним трансляцию с получением файла листинга(nasm -f elf -l lab7-2.lst lab7-2.asm) (рис. 3.20)

```
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2-1.lst lab7-2-1.asm
lab7-2-1.asm:24: error: invalid combination of opcode and operands
oskalashnikova@dk6n62:~/work/arch-pc/lab07$ █
```

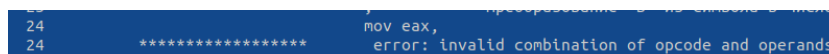
Рис. 3.20: Трансляция с получением файла листинга

Мы видим ошибку и создаётся только файл листинга (рис. 3.21)

```
in_out.asm
*lab7-1
*lab7-1-2
  lab7-1-2.asm
  lab7-1-2.o
*lab7-1-3
  lab7-1-3.asm
  lab7-1-3.o
  lab7-1.asm
  lab7-1.o
*lab7-2
  lab7-2-1.asm
  lab7-2-1.lst
  lab7-2.asm
  lab7-2.lst
  lab7-2.o
```

Рис. 3.21: Созданный файл

Зайдя в листинг, мы видим что в нём создаётся строка, предупреждающая об ошибке (рис. 3.22)



```
24      ; преобразование в int-значение в меню
24      *****
24      mov eax,
24      error: invalid combination of opcode and operands
```

Рис. 3.22: Файл листинга

### 3.3 Задание для самостоятельной работы

Задание 1: 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.

Вносим изменения в программу Листинга 7.3 для нахождения минимума из трёх введённых с клавиатуры переменных. Используем jl (Переход если a меньше b). (рис. 3.23), (рис. 3.24)

```

/home/oskalashnikova/work/arch-pc
%include 'in_out.asm'

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

```

Рис. 3.23: Изменение кода 1

```

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

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

```

Рис. 3.24: Изменение кода 2

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab7-3.asm`, `ld -m elf_i386 -o lab7-3 lab7-3.o`, запуск: `./lab7-3`), вводим значения из своего варианта (рис. 3.25), (рис. 3.26)

```
oskashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
oskashnikova@dk6n62:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
oskashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-3
Введите A 45
Введите B 67
Введите C 15
Наименьшее число: 15
oskashnikova@dk6n62:~/work/arch-pc/lab07$
```

Рис. 3.25: Создание и запуск исполняемого файла

7 45,67,15

Рис. 3.26: Значения в варианте

Текст программы:

```
%include 'in_out.asm'
```

```
section .data
```

```
msg db "Наименьшее число: ",0h
```

```
msg1 db "Введите A ",0h
```

```
msg2 db "Введите B ",0h
```

```
msg3 db "Введите C ",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:
cmp ecx,[B]
jl fin
mov ecx,[B]
mov [min],ecx

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

```

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

Пишем программу, которая вычисляет значение функции:  $ba$  если  $x=a$  и  $a+x$

если  $x \neq a$  (рис. 3.27), (рис. 3.28), (рис. 3.29)

$$7 \quad \begin{cases} 6a, & x = a \\ a + x, & x \neq a \end{cases} \quad (1;1) \quad (2;1)$$

Рис. 3.27: Вариант 7

```
/home/oskalashnikova/work/arch-pc
#include 'in_out.asm'

section .data
msg1 db 'Введите x: ',0h
msg2 db 'Введите a: ',0h
msg db 'answer ',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
```

Рис. 3.28: Текст программы 1

```
mov eax, [a]
mov ecx, [x]
cmp eax,ecx
je ifravn
add eax,ecx
jmp fin

ifravn:
mov ebx,6
mul ebx

fin:
mov ebx,eax
mov eax, msg
call sprint
mov eax,ebx
call iprintLF.
call quit.
```

Рис. 3.29: Текст программы 2

Создаём исполняемый файл и запускаем его (компиляция: `nasm -f elf lab7-4.asm`, `ld -m elf_i386 -o lab7-4 lab7-4.o`, запуск: `./lab7-4`), вводим значения из своего варианта (рис. 3.30)

```
oskashnikova@dk6n62:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
oskashnikova@dk6n62:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
oskashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 1
Введите a: 1
answer 6
oskashnikova@dk6n62:~/work/arch-pc/lab07$ ./lab7-4
Введите x: 2
Введите a: 1
answer 3
oskashnikova@dk6n62:~/work/arch-pc/lab07$
```

Рис. 3.30: Создание и запуск исполняемого файла

```
%include 'in_out.asm'
```

```
section .data
```

```
msg1 db 'Введите x: ',0h
```

```
msg2 db 'Введите a: ',0h
```

```
msg db 'answer ',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 eax, [a]
mov ecx, [x]
cmp eax,ecx
je ifravn
add eax,ecx
jmp fin
```

```
ifravn:
mov ebx,6
mul ebx
```

```
fin:
mov ebx,eax
```

```
mov eax, msg  
call sprint  
mov eax, ebx  
call iprintLF  
call quit
```

## 4 Выводы

В ходе данной лабораторной работы мы изучили команды условного и безусловного перехода, приобрели навыков написания программ с использованием переходов, познакомились с назначением и структурой файла листинга.