

Лабораторная работа № 7

**Команды безусловного и условного переходов в Nasm.
Программирование ветвлений**

Глушенок Анна Александровна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Задание 1. Реализация переходов в NASM	6
2.2	Задание 2. Изучение структуры файла листинга	11
3	Задания для самостоятельной работы	15
4	Выводы	19

Список иллюстраций

2.1	создание файла для работы	6
2.2	Программа с использованием инструкции jmp	7
2.3	Результат работы программы	7
2.4	Программа с использованием инструкции jmp	8
2.5	Результат работы программы	8
2.6	Внесение изменений в программу	9
2.7	Результат работы программы	9
2.8	Программа вывода наибольшей переменной	10
2.9	Результат работы программы	11
2.10	Создание файла листинга	11
2.11	Ознакомление с файлом листинга	12
2.12	Удаление операнда	13
2.13	Транслирование с получением листинга	13
2.14	Полученный листинг с ошибкой	14
3.1	Написание программы	16
3.2	Проверка работы программы	16
3.3	Написание программы	17
3.4	Проверка работы программы	18

Список таблиц

1 Цель работы

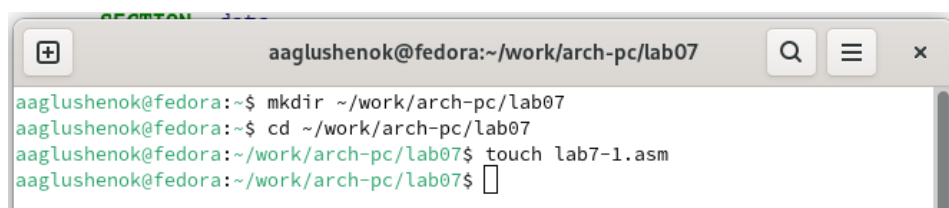
Изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Задание 1. Реализация переходов в NASM

1. Создайте каталог для программ лабораторной работы № 7, перейдите в него и создайте файл lab7-1.asm.

Создаем указанный каталог, переходим в него и создаем внутри файл lab7-1.asm (команда mkdir).

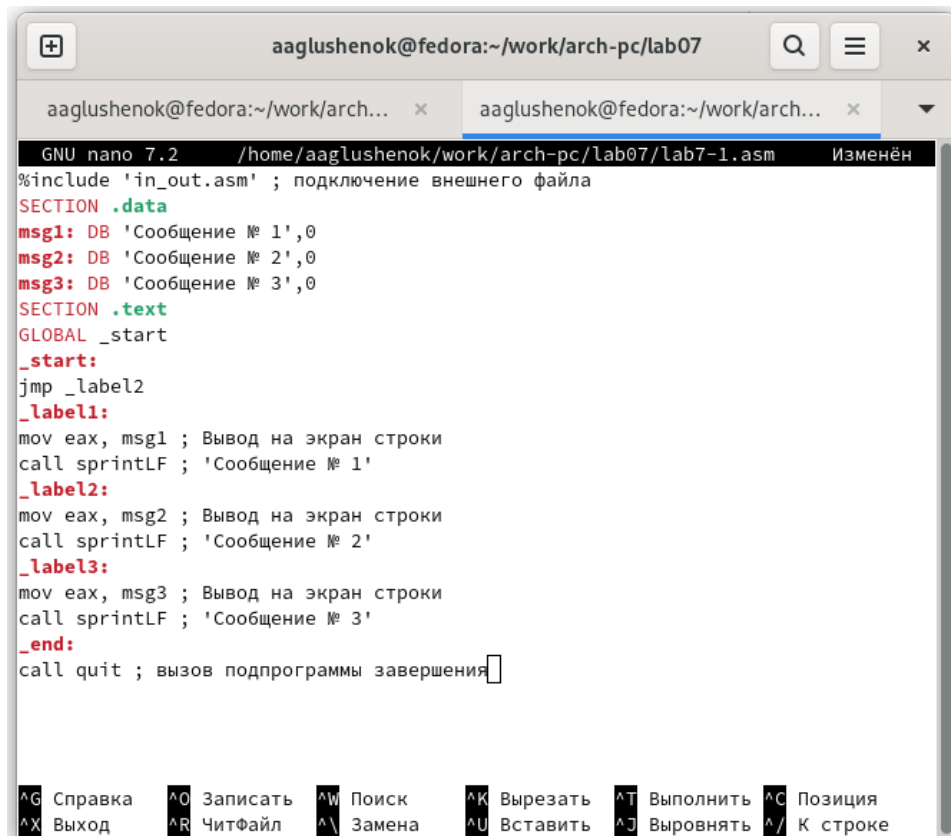
A screenshot of a terminal window with a title bar that reads 'aaglushenok@fedora:~/work/arch-pc/lab07'. The terminal shows the following commands and their outputs: 'mkdir ~/work/arch-pc/lab07', 'cd ~/work/arch-pc/lab07', and 'touch lab7-1.asm'. The prompt changes from '~\$' to '~/work/arch-pc/lab07\$' after each command. The window has standard Linux window controls (minimize, maximize, close) and search and menu icons in the title bar.

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

Рис. 2.1: создание файла для работы

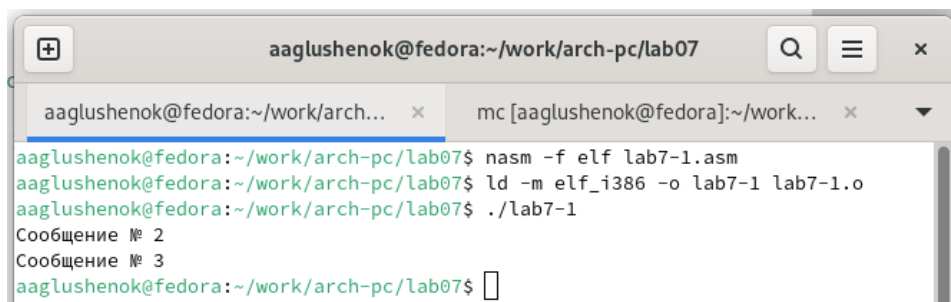
2. Рассмотрите пример программы с использованием инструкции jmp. Введите в файл lab7-1.asm текст программы из листинга 7.1. Создайте исполняемый файл и запустите его.

Вводим в файл lab7-1.asm текст программы из листинга 7.1. (Программа с использованием инструкции jmp), создаем исполняемый файл и запускаем его.



```
GNU nano 7.2 /home/aaglushenok/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. 2.2: Программа с использованием инструкции jmp



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

Рис. 2.3: Результат работы программы

3. Измените программу таким образом, чтобы она выводила сначала “Сообщение № 2”, потом “Сообщение № 1” и завершала работу. Создайте исполняемый файл и проверьте его работу.

Меняем программу в соответствии с листингом 7.2. (Программа с использованием инструкции jmp): в текст программы после вывода сообщения № 2 добавля-

ем инструкцию `jmp` с меткой `_label1`, а после вывода сообщения № 1 добавляем инструкцию `jmp` с меткой `_end`. Создаем исполняемый файл и проверяем его работу.

```

GNU nano 7.2 /home/aaglushenok/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'
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. 2.4: Программа с использованием инструкции `jmp`

```

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

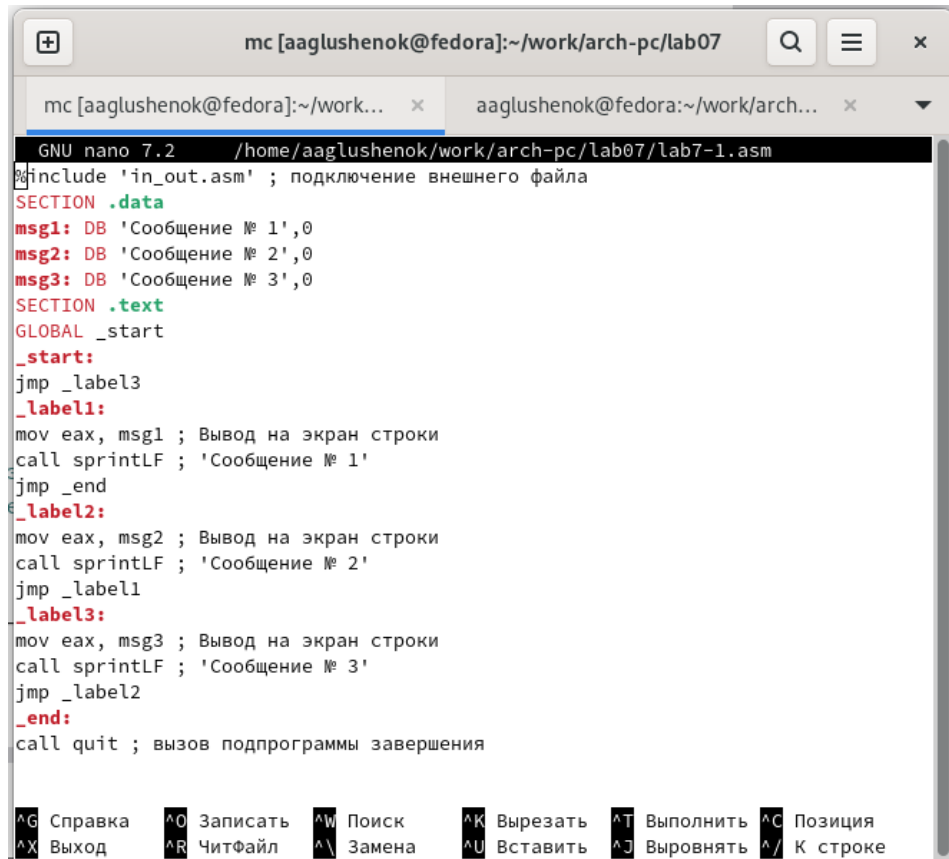
```

Рис. 2.5: Результат работы программы

4. Измените текст программы, добавив или изменив инструкции `jmp`, чтобы программа выводила сначала “Сообщение № 3, потом”Сообщение № 2”,

“Сообщение № 1” и завершала работу.

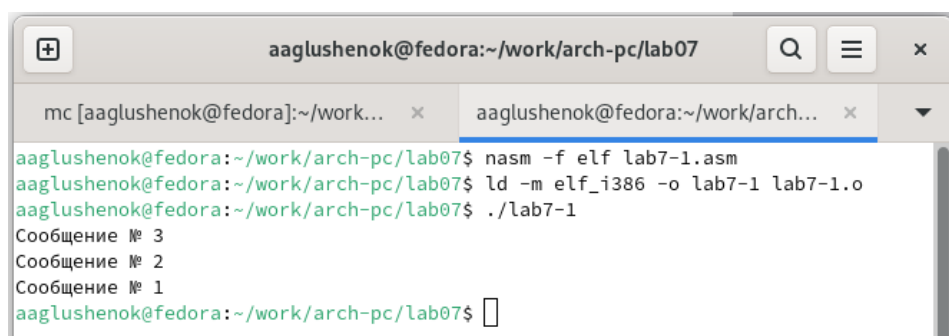
Вносим изменения в программу, добавляем инструкции `jmp` и меняем их метки. Создаем исполняемый файл и проверяем результат работы программы.



```
mc [aaglushmanok@fedora]:~/work/arch-pc/lab07
GNU nano 7.2 /home/aaglushmanok/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
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выворнуть ^_/ К строке

Рис. 2.6: Внесение изменений в программу

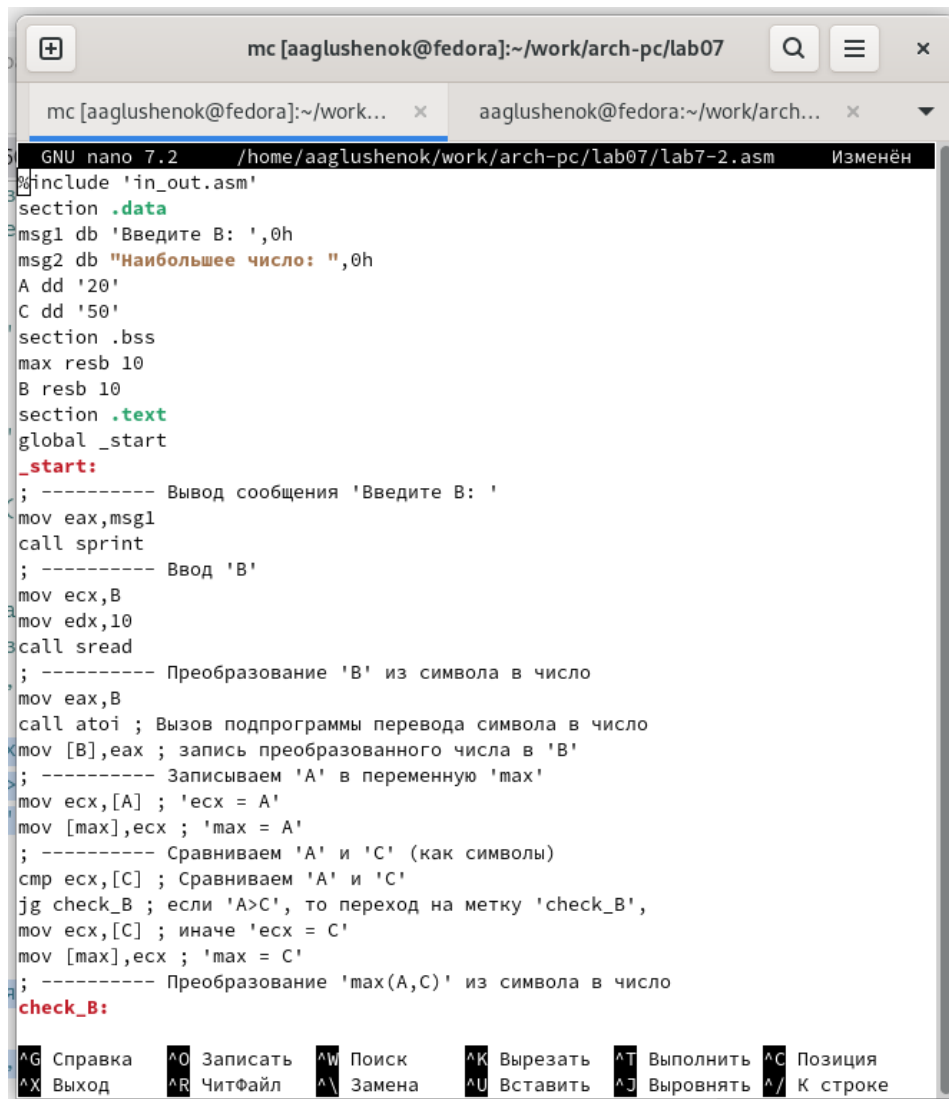


```
aaglushmanok@fedora:~/work/arch-pc/lab07
mc [aaglushmanok@fedora]:~/work/arch-pc/lab07
aaglushmanok@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
aaglushmanok@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
aaglushmanok@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
aaglushmanok@fedora:~/work/arch-pc/lab07$
```

Рис. 2.7: Результат работы программы

5. Создайте файл lab7-2.asm в каталоге ~/work/arch-pc/lab07. Внимательно изучите текст программы из листинга 7.3 и введите в lab7-2.asm. Создайте исполняемый файл и проверьте его работу для разных значений В.

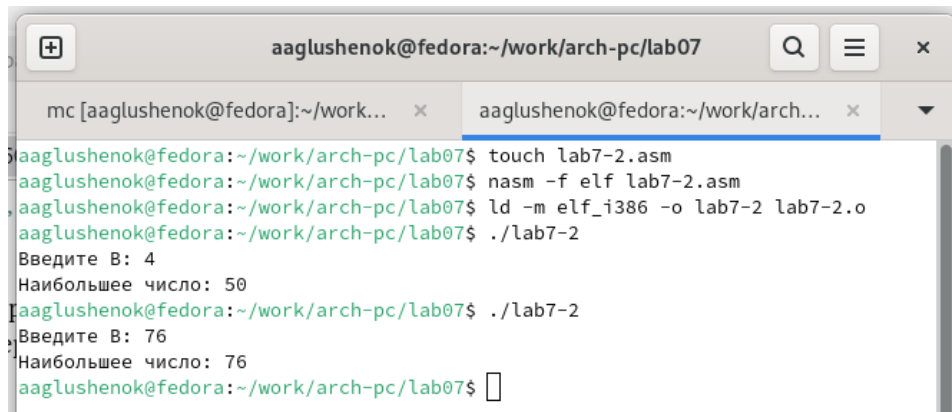
Вводим в файл lab7-2.asm текст из листинга 7.3. (Программа, которая определяет и выводит на экран наибольшую из 3 переменных: А,В и С). Создаем исполняемый файл и проверяем его работу для разных значений В.



```
mc [aaglushenok@fedora]:~/work/arch-pc/lab07
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab07/lab7-2.asm
%include 'in_out.asm'
section .data
msg1 db 'Введите В: ',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 В: '
mov eax,msg1
call sprint
; ----- Ввод 'В'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'В' из символа в число
mov eax,B
call atoi ; Вызов подпрограммы перевода символа в число
mov [B],eax ; запись преобразованного числа в 'В'
; ----- Записываем 'А' в переменную 'max'
mov ecx,[A] ; 'ecx = А'
mov [max],ecx ; 'max = А'
; ----- Сравниваем 'А' и 'С' (как символы)
cmp ecx,[C] ; Сравниваем 'А' и 'С'
jg check_B ; если 'А>С', то переход на метку 'check_B',
mov ecx,[C] ; иначе 'ecx = С'
mov [max],ecx ; 'max = С'
; ----- Преобразование 'max(А,С)' из символа в число
check_B:
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

Рис. 2.8: Программа вывода наибольшей переменной



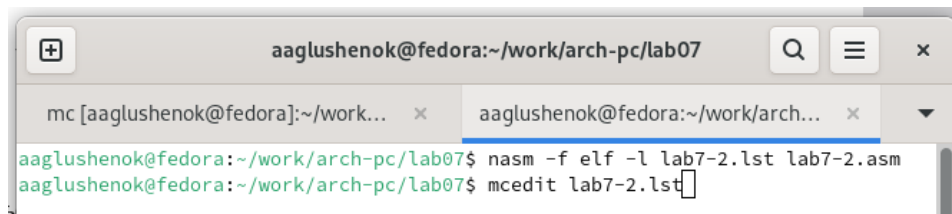
```
aaglushmanok@fedora:~/work/arch-pc/lab07
mc [aaglushmanok@fedora]:~/work... x aaglushmanok@fedora:~/work/arch... x
aaglushmanok@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
aaglushmanok@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
aaglushmanok@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
aaglushmanok@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 4
Наибольшее число: 50
aaglushmanok@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 76
Наибольшее число: 76
aaglushmanok@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Результат работы программы

2.2 Задание 2. Изучение структуры файла листинга

6. Создайте файл листинга для программы из файла lab7-2.asm. Откройте файл листинга lab7-2.lst с помощью любого текстового редактора. Внимательно ознакомьтесь с форматом и содержимым листинга. Подробно объясните содержимое трёх строк файла листинга по выбору

Создаем файл листинга для программы из файла lab7-2.asm, указав ключ `-l` и имя файла в командной строке.



```
aaglushmanok@fedora:~/work/arch-pc/lab07
mc [aaglushmanok@fedora]:~/work... x aaglushmanok@fedora:~/work/arch... x
aaglushmanok@fedora:~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
aaglushmanok@fedora:~/work/arch-pc/lab07$ mcedit lab7-2.lst
```

Рис. 2.10: Создание файла листинга

```

1 %include 'in_out.asm'
2 <1> ;----- slen -----
3 <1> ; функция вычисления длины сообщения
4 <1> slen:.....
5 00000000 53 <1> push ebx.....
6 00000001 89C3 <1> mov ebx, eax.....
7 <1>.....
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 <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> ;----- 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 B801000000 <1> mov ebx, 1

```

Рис. 2.11: Ознакомление с файлом листинга

Строка 29: “00000018” - адрес в сегменте кода, “89C2” - машинный код, “mov edx, eax” - копирование значения из регистра eax в регистр edx.

Строка 34: “00000022” - адрес в сегменте кода, “B804000000” -машинный код, “mov eax,4” -присвоение переменной eax значения 4.

- Откройте файл с программой lab7-2.asm и в любой инструкции с двумя операндами удалите один операнд. Выполните трансляцию с получением файла листинга. Какие выходные файлы создаются в этом случае?

Открываем файл и удаляем один из операндов. Выполняем трансляцию файла

и изучаем файл листинга с ошибкой. При трансляции файла, выдается ошибка, но создаются исполняемые файлы lab7-2 и lab7-2.lst.

```

GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab07/lab7-2.asm Изменён
%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] ; 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
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^_ К строке

```

Рис. 2.12: Удаление операнда

```

aaglushenok@fedora:~/work/arch-pc/lab07
mc [aaglushenok@fedora]:~/work/ar... x aaglushenok@fedora:~/work/arch-p... x
aaglushenok@fedora:~/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
aaglushenok@fedora:~/work/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2 lab7-2.asm lab7-2.lst
aaglushenok@fedora:~/work/arch-pc/lab07$

```

Рис. 2.13: Транслирование с получением листинга

```

aaglushmanok@fedora:~/work/arch-pc/lab07 — mcedit lab7-2.lst
mc [aaglushmanok@fedora]:~ x aaglushmanok@fedora:~/work/arch-pc/lab07 ... x
lab7-2.lst [-----] 0 L: [ 97+ 0 97/226] * (5959/14546b) 0032 0x020 [*][X]
96 00000076 E894FFFFFF <1> call sprint...
97 00000078 58 <1> pop eax...
98 0000007C 83F900 <1> cmp ecx, 0...
99 0000007F 75F2 <1> jnz printLoop..
100 <1> .
101 00000081 5E <1> pop esi...
102 00000082 5A <1> pop edx...
103 00000083 59 <1> pop ecx...
104 00000084 58 <1> pop eax.....
105 00000085 C3 <1> ret
106 <1> .
107 <1> .
108 <1> ;----- iprintLF -----
109 <1> ; Функция вывода на экран чисел в формате ASCII
110 <1> ; входные данные: mov eax,<int>
111 <1> iprintLF:
112 00000086 E8C9FFFFFF <1> call iprint.....
113 <1> .
114 0000008B 50 <1> push eax.....
115 0000008C B80A000000 <1> mov eax, 0Ah.....
116 00000091 50 <1> push eax.....
117 00000092 89E0 <1> mov eax, esp.....
118 00000094 E876FFFFFF <1> call sprint.....
119 00000099 58 <1> pop eax.....
120 0000009A 58 <1> pop eax.....
121 0000009B C3 <1> ret
122 <1> .
123 <1> ;----- atoi -----
124 <1> ; Функция преобразования ascii-код символа в целое число
125 <1> ; входные данные: mov eax,<int>
126 <1> atoi:
127 0000009C 53 <1> push ebx.....
128 0000009D 51 <1> push ecx.....
129 0000009E 52 <1> push edx.....
130 0000009F 56 <1> push esi.....
131 000000A0 89C6 <1> mov esi, eax.....
132 000000A2 B800000000 <1> mov eax, 0.....
133 000000A7 B900000000 <1> mov ecx, 0.....
134 <1> .
135 <1> .multiplyLoop:
136 000000AC 31DB <1> xor ebx, ebx.....
137 000000AE 8A1C0E <1> mov bl, [esi+ecx]
138 000000B1 80FB30 <1> cmp bl, 48..
139 000000B4 7C14 <1> jl .finished.
140 000000B6 80FB39 <1> cmp bl, 57..
1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Пер-ить 7 Поиск 8 Удалить 9 МенюМС 10 Выход

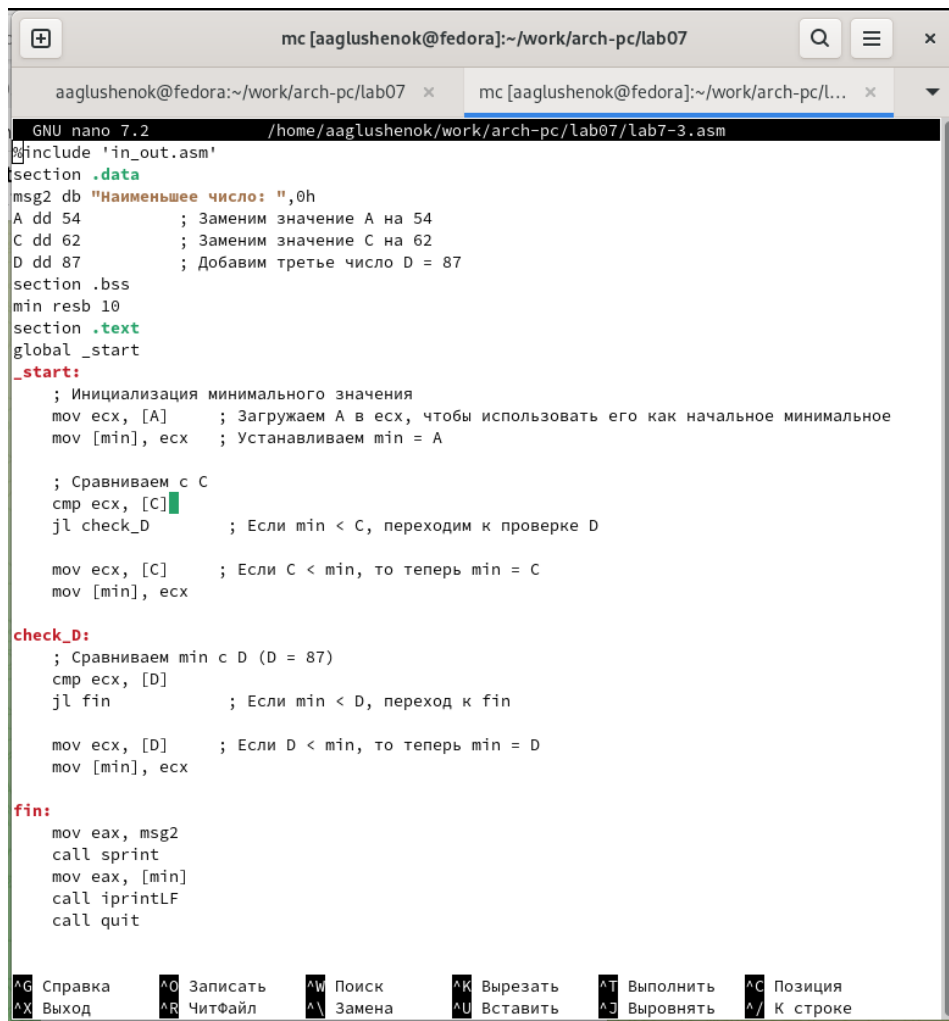
```

Рис. 2.14: Полученный листинг с ошибкой

3 Задания для самостоятельной работы

1. Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных - 54, 62, 87 (вариант 5). Создайте исполняемый файл и проверьте его работу.

Создаем файл lab7-3.asm, пишем программу, аналогичную предыдущим. Создаем исполняемый файл, проверяем корректность работы программы (ответ 54 - верный).



The screenshot shows a terminal window with the nano 7.2 editor open. The file being edited is `lab7-3.asm` located at `/home/aaglushenok/work/arch-pc/lab07/`. The code is written in assembly language and includes comments in Russian. It defines a data section with a message `msg2` and initializes variables `A`, `C`, and `D`. The `.text` section starts at `_start` and contains logic to find the minimum of three numbers. It uses `mov`, `cmp`, and `jnl` instructions. A `check_D` label handles a specific case for variable `D`. The `fin` label prints the result using `sprint` and `iprintLF` before calling `quit`. At the bottom, a keyboard shortcut legend is visible.

```
GNU nano 7.2 /home/aaglushenok/work/arch-pc/lab07/lab7-3.asm
#include 'in_out.asm'
section .data
msg2 db "Наименьшее число: ",0h
A dd 54 ; Заменяем значение A на 54
C dd 62 ; Заменяем значение C на 62
D dd 87 ; Добавим третье число D = 87
section .bss
min resb 10
section .text
global _start
_start:
; Инициализация минимального значения
mov ecx, [A] ; Загружаем A в ecx, чтобы использовать его как начальное минимальное
mov [min], ecx ; Устанавливаем min = A

; Сравниваем с C
cmp ecx, [C]
jnl check_D ; Если min < C, переходим к проверке D

mov ecx, [C] ; Если C < min, то теперь min = C
mov [min], ecx

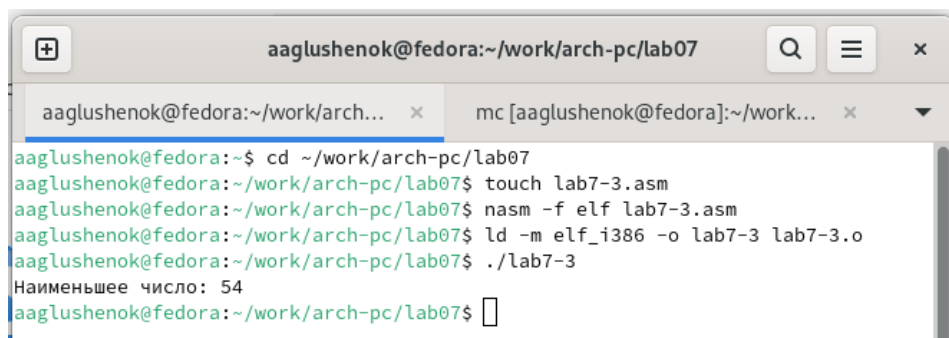
check_D:
; Сравниваем min с D (D = 87)
cmp ecx, [D]
jnl fin ; Если min < D, переход к fin

mov ecx, [D] ; Если D < min, то теперь min = D
mov [min], ecx

fin:
mov eax, msg2
call sprint
mov eax, [min]
call iprintLF
call quit

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R Читфайл ^\ Замена ^U Вставить ^_ Выровнять ^/ К строке
```

Рис. 3.1: Написание программы



The screenshot shows a terminal window where the assembly program is compiled and executed. The user navigates to the directory `~/work/arch-pc/lab07` and runs the following commands: `touch lab7-3.asm`, `nasm -f elf lab7-3.asm`, `ld -m elf_i386 -o lab7-3 lab7-3.o`, and `./lab7-3`. The output of the program is `Наименьшее число: 54`.

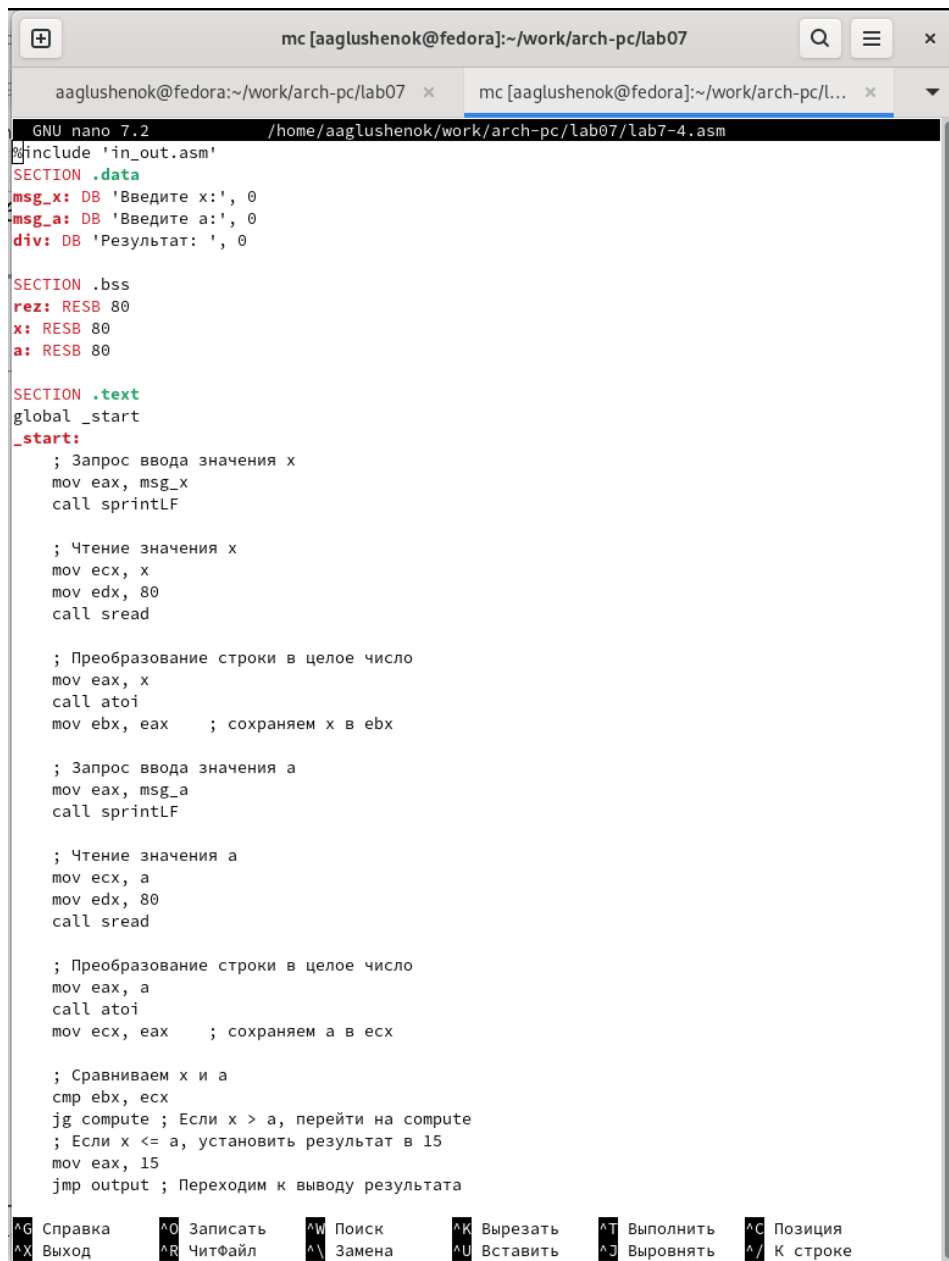
```
aaglushenok@fedora:~$ cd ~/work/arch-pc/lab07
aaglushenok@fedora:~/work/arch-pc/lab07$ touch lab7-3.asm
aaglushenok@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
aaglushenok@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
aaglushenok@fedora:~/work/arch-pc/lab07$ ./lab7-3
Наименьшее число: 54
aaglushenok@fedora:~/work/arch-pc/lab07$
```

Рис. 3.2: Проверка работы программы

2. Напишите программу, которая для введенных с клавиатуры значений x и a вычисляет значение функции $f(x) = 2(x-a)$, $x > a$; 15 , $x \leq a$; (вариант 5) и

выводит результат вычислений. Создайте исполняемый файл и проверьте его работу для значений $x=1$, $a=2$ и $x=2$, $a=1$.

Создаем файл lab7-4.asm, пишем программу, аналогичную программам из лабораторных работ 6 и 7. Создаем исполняемый файл, проверяем корректность его работы (ответы 32 и 30 - верные).



```
GNU nano 7.2 /home/aaglushmanok/work/arch-pc/lab07/lab7-4.asm
#include 'in_out.asm'
SECTION .data
msg_x: DB 'Введите x:', 0
msg_a: DB 'Введите a:', 0
div: DB 'Результат: ', 0

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

SECTION .text
global _start
_start:
; Запрос ввода значения x
mov eax, msg_x
call sprintf

; Чтение значения x
mov ecx, x
mov edx, 80
call sread

; Преобразование строки в целое число
mov eax, x
call atoi
mov ebx, eax ; сохраняем x в ebx

; Запрос ввода значения a
mov eax, msg_a
call sprintf

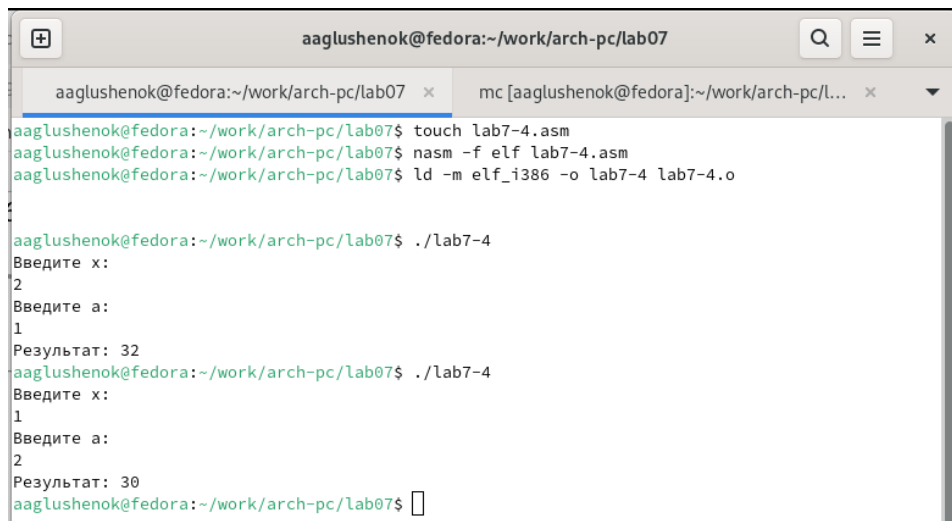
; Чтение значения a
mov ecx, a
mov edx, 80
call sread

; Преобразование строки в целое число
mov eax, a
call atoi
mov ecx, eax ; сохраняем a в ecx

; Сравниваем x и a
cmp ebx, ecx
jg compute ; Если x > a, перейти на compute
; Если x <= a, установить результат в 15
mov eax, 15
jmp output ; Переходим к выводу результата

^G Справка      ^O Записать     ^W Поиск        ^K Вырезать     ^T Выполнить    ^C Позиция
^X Выход        ^R ЧитФайл     ^\ Замена       ^U Вставить     ^J Вывод        ^_ К строке
```

Рис. 3.3: Написание программы



The image shows a terminal window with a title bar containing the username 'aaglushenok@fedora' and the current directory '~/work/arch-pc/lab07'. There are also search, menu, and close icons in the title bar. The terminal has two tabs: the active one is 'aaglushenok@fedora:~/work/arch-pc/lab07' and the other is 'mc [aaglushenok@fedora]:~/work/arch-pc/l...'. The terminal content shows the following commands and output:

```
aaglushenok@fedora:~/work/arch-pc/lab07$ touch lab7-4.asm
aaglushenok@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
aaglushenok@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o

aaglushenok@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x:
2
Введите a:
1
Результат: 32
aaglushenok@fedora:~/work/arch-pc/lab07$ ./lab7-4
Введите x:
1
Введите a:
2
Результат: 30
aaglushenok@fedora:~/work/arch-pc/lab07$
```

Рис. 3.4: Проверка работы программы

4 Выводы

В ходе выполнения лабораторной работы мне удалось изучить команды условного и безусловного переходов, приобрести навыки написания программ с использованием переходов, а так же познакомиться с назначением и структурой файла листинга.