

Отчёт по лабораторной работе №7

Ярослав Антонович Меркулов

Содержание

1	Цель работы	4
2	Выполнение лабораторной работы	5
2.1	Реализация переходов в NASM	5
2.2	Изучение структуры файлы листинга	10
3	Выполнение самостоятельной работы	12
4	Выводы	15

Список иллюстраций

2.1	Создание каталога, переход в него, создание файла	5
2.2	Введённый текст программы	6
2.3	Работа программы	6
2.4	Изменённый файл	7
2.5	Работа новой программы	7
2.6	Программа для обратного вывода	8
2.7	Работа программы	8
2.8	Файл lab7-2.asm	9
2.9	Работа программы	9
2.10	Создание файла листинга	10
2.11	Файл листинга	11
2.12	Ошибка	11
3.1	Готовый lab7-3.asm	12
3.2	Работа программы	13
3.3	Готовый lab7-4.asm	13
3.4	Работа программы	14

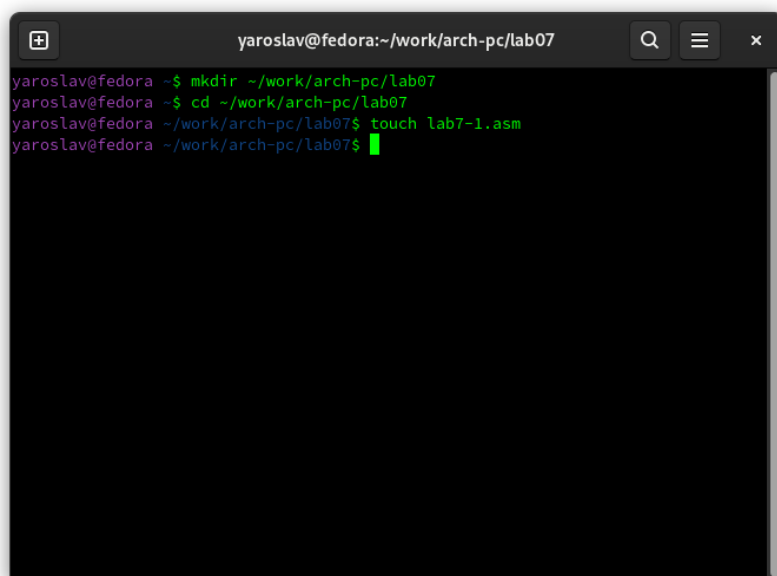
1 Цель работы

Изучить команды условного и безусловного переходов, приобрести навыки написания программ с переходами, познакомиться с назначением и структурой файла листинга

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

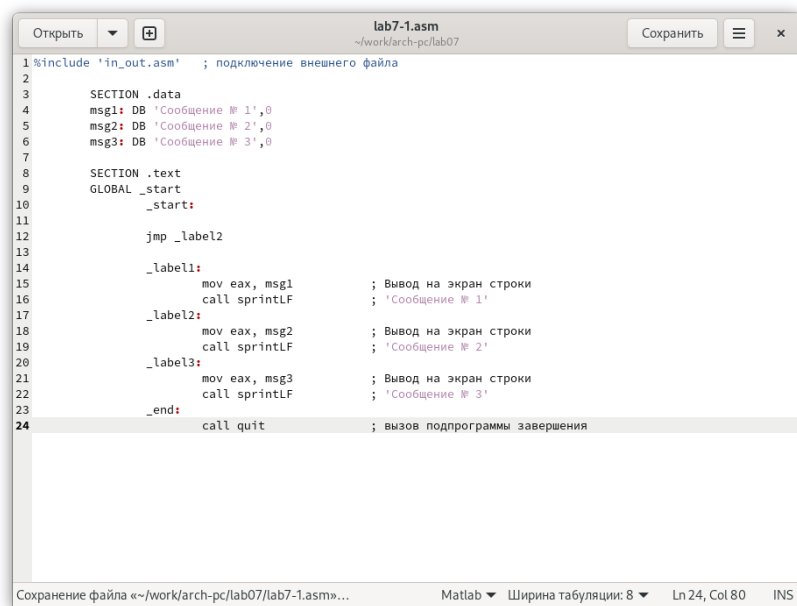
1. Создаём каталог для лабораторной работы, переходим в него и создаём файл lab7-1.asm.



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

Рис. 2.1: Создание каталога, переход в него, создание файла

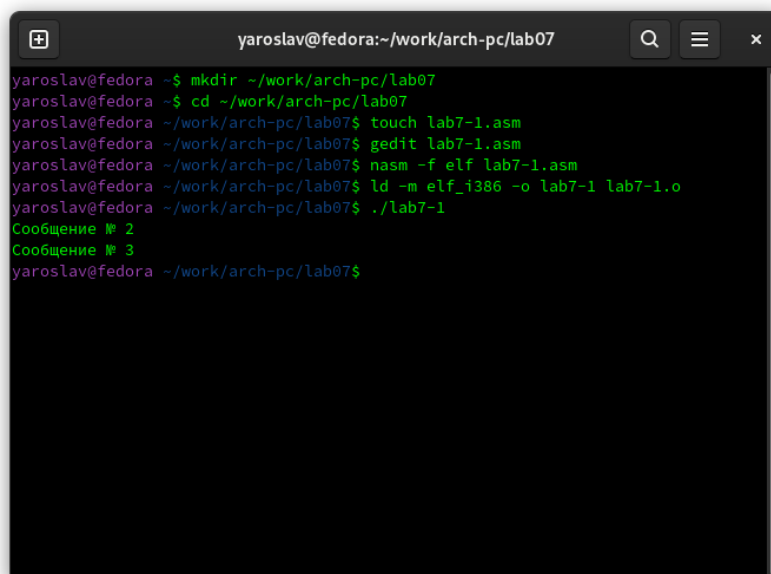
2. Вводим текст программы из листинга 7.1.



```
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 jmp _label2
13
14 _label1:
15 mov eax, msg1 ; Вывод на экран строки
16 call sprintf ; 'Сообщение № 1'
17
18 _label2:
19 mov eax, msg2 ; Вывод на экран строки
20 call sprintf ; 'Сообщение № 2'
21
22 _label3:
23 mov eax, msg3 ; Вывод на экран строки
24 call sprintf ; 'Сообщение № 3'
25
26 _end:
27 call quit ; вызов подпрограммы завершения
```

Рис. 2.2: Введённый текст программы

3. Создаём исполняемый файл и запускаем.



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

Рис. 2.3: Работа программы

4. Изменяем текст программы, создаём исполняемый файл и запускаем.

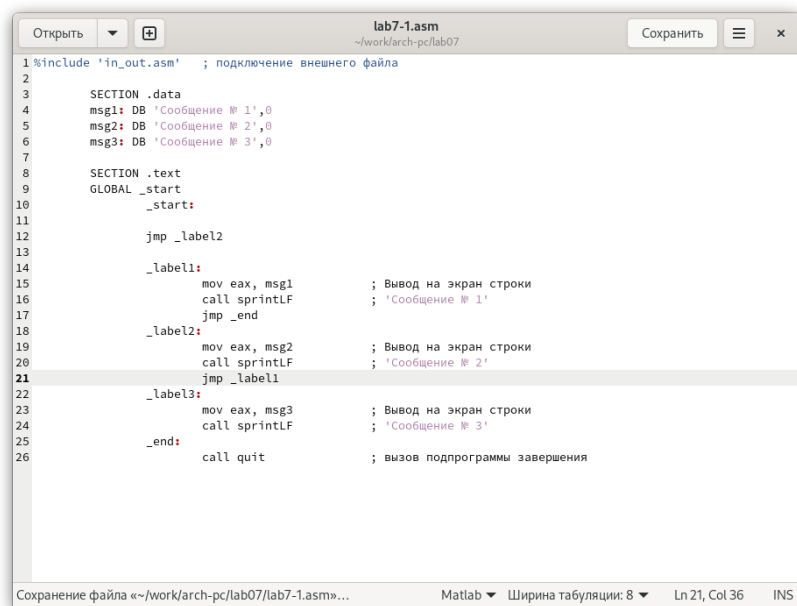


Рис. 2.4: Изменённый файл

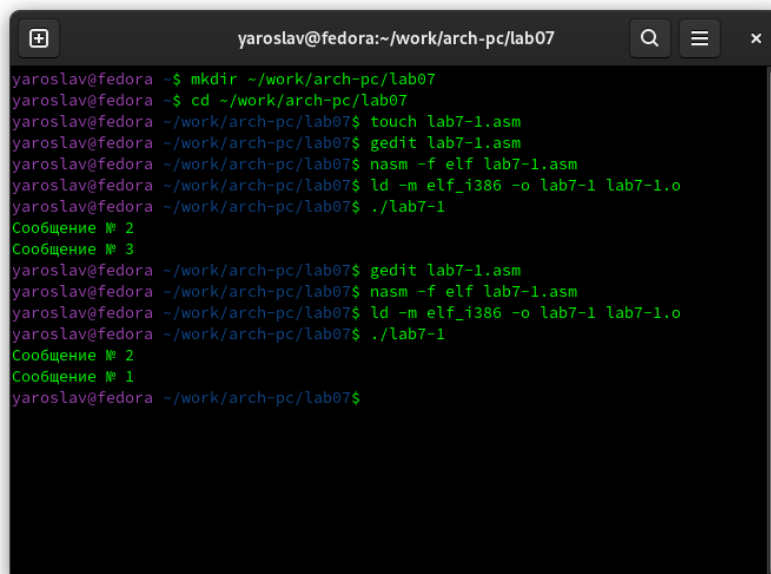


Рис. 2.5: Работа новой программы

5. Меняем программу так, чтоб сообщения выводились в обратном порядке.

```

1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4 msg1: DB 'Сообщение № 1',0
5 msg2: DB 'Сообщение № 2',0
6 msg3: DB 'Сообщение № 3',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12     jmp _label3
13
14     _label1:
15         mov eax, msg1      ; Вывод на экран строки
16         call sprintf       ; 'Сообщение № 1'
17         jmp _end
18     _label2:
19         mov eax, msg2      ; Вывод на экран строки
20         call sprintf       ; 'Сообщение № 2'
21         jmp _label1
22     _label3:
23         mov eax, msg3      ; Вывод на экран строки
24         call sprintf       ; 'Сообщение № 3'
25         jmp _label2
26     _end:
27         call quit          ; вызов подпрограммы завершения

```

Рис. 2.6: Программа для обратного вывода

6. Создаём исполняемый файл и запускаем.

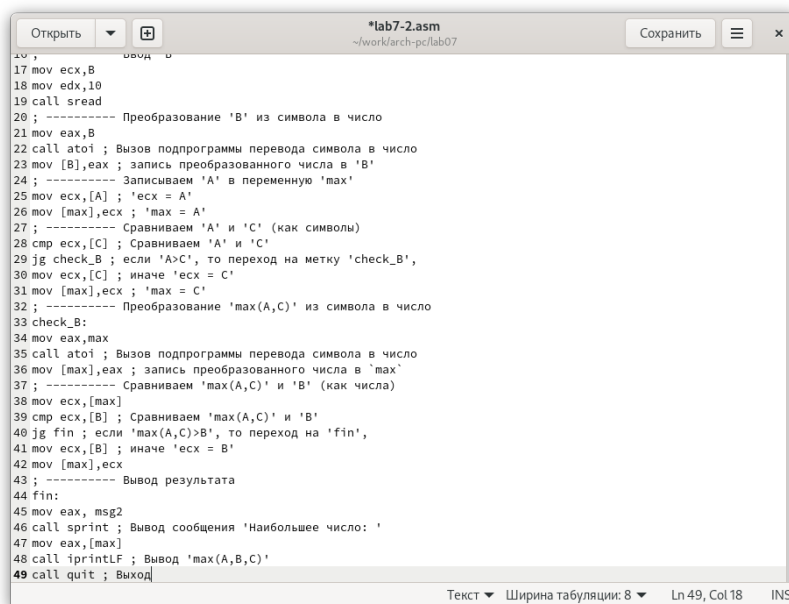
```

yaroslav@fedora: ~/work/arch-pc/lab07
yaroslav@fedora ~$ mkdir ~/work/arch-pc/lab07
yaroslav@fedora ~$ cd ~/work/arch-pc/lab07
yaroslav@fedora ~/work/arch-pc/lab07$ touch lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ gedit lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
yaroslav@fedora ~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
yaroslav@fedora ~/work/arch-pc/lab07$ gedit lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
yaroslav@fedora ~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
yaroslav@fedora ~/work/arch-pc/lab07$ gedit lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
yaroslav@fedora ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
yaroslav@fedora ~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
yaroslav@fedora ~/work/arch-pc/lab07$

```

Рис. 2.7: Работа программы

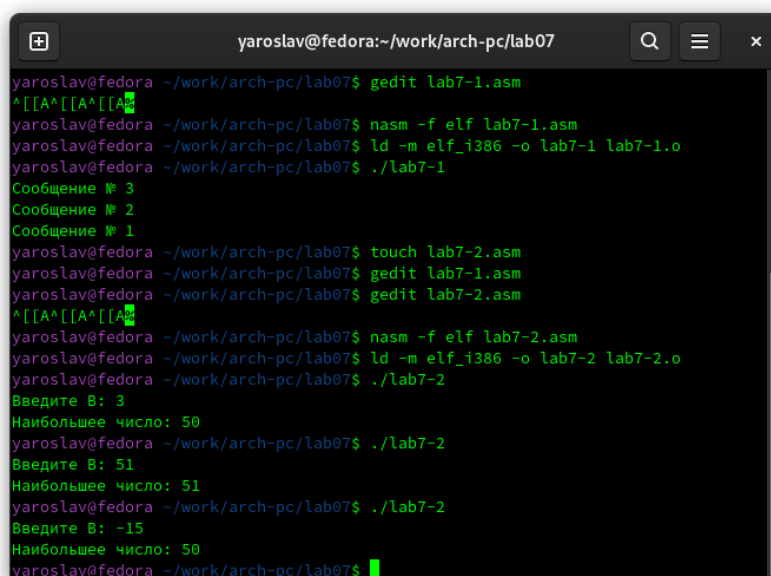
7. Создаём файл lab7-2.asm и вводим листинг 7.3 в него.



```
17 mov ecx,8
18 mov edx,10
19 call sread
20 ; ----- Преобразование 'B' из символа в число
21 mov eax,8
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)
38 mov ecx,[max]
39 cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
40 jg fin ; если 'max(A,C)>B', то переход на 'fin',
41 mov ecx,[B] ; иначе 'ecx = B'
42 mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 mov eax, msg2
46 call sprintf ; Вывод сообщения 'Наибольшее число: '
47 mov eax,[max]
48 call sprintf ; Вывод 'max(A,B,C)'
49 call quit ; Выход
```

Рис. 2.8: Файл lab7-2.asm

8. Создаём исполняемый файл и тестируем с разными числами.

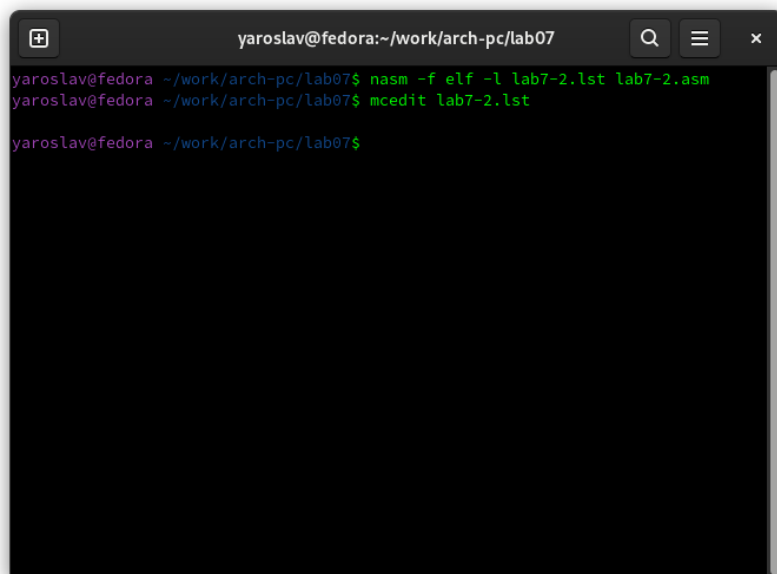


```
yaroslav@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm
yaroslav@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
yaroslav@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-1 lab7-1.o
yaroslav@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
yaroslav@fedora:~/work/arch-pc/lab07$ touch lab7-2.asm
yaroslav@fedora:~/work/arch-pc/lab07$ gedit lab7-1.asm
yaroslav@fedora:~/work/arch-pc/lab07$ gedit lab7-2.asm
yaroslav@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
yaroslav@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-2 lab7-2.o
yaroslav@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 3
Наибольшее число: 50
yaroslav@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 51
Наибольшее число: 51
yaroslav@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: -15
Наибольшее число: 50
yaroslav@fedora:~/work/arch-pc/lab07$
```

Рис. 2.9: Работа программы

2.2 Изучение структуры файлы листинга

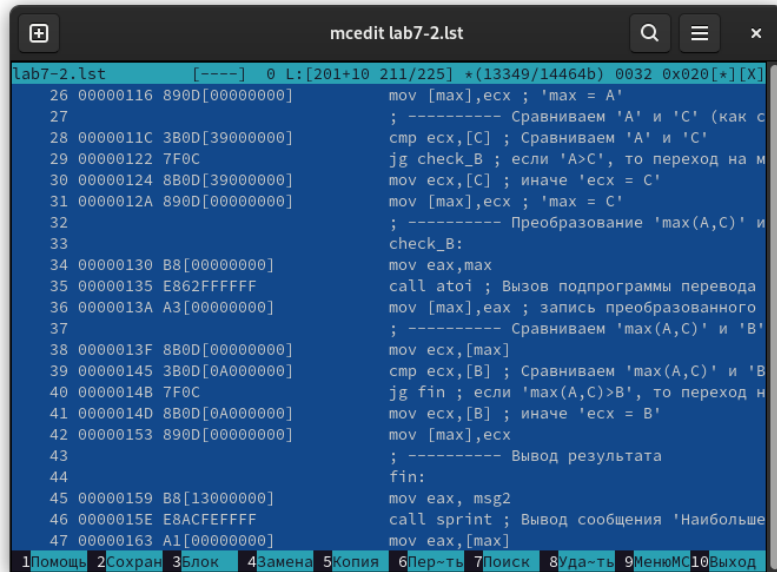
9. Создаём файл листинга и открываем с помощью mcedit.

A terminal window titled 'yaroslav@fedora:~/work/arch-pc/lab07' with search, menu, and close icons. It shows three commands being executed: 'nasm -f elf -l lab7-2.lst lab7-2.asm', 'mcedit lab7-2.lst', and a final prompt 'yaroslav@fedora ~/work/arch-pc/lab07\$'.

```
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
yaroslav@fedora ~/work/arch-pc/lab07$ mcedit lab7-2.lst
yaroslav@fedora ~/work/arch-pc/lab07$
```

Рис. 2.10: Создание файла листинга

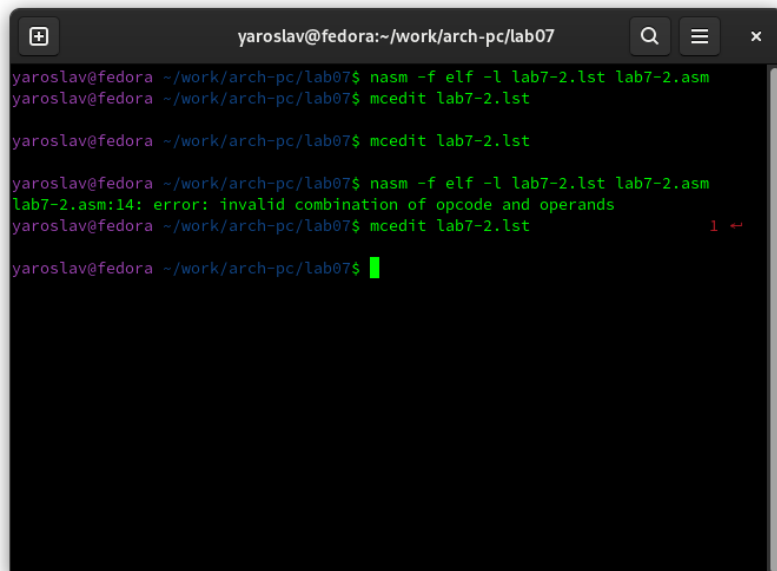
10. Просматриваем файл листинга. В каждой строке мы видим номер строки, адрес, машинный код и исходный текст программы.



```
lab7-2.lst [----] 0 L:[201+10 211/225] *(13349/14464b) 0032 0x020[*][X]
26 00000116 890D[00000000]    mov [max],ecx ; 'max = A'
27                               ; ----- Сравниваем 'A' и 'C' (как с
28 0000011C 3B0D[39000000]    cmp ecx,[C] ; Сравниваем 'A' и 'C'
29 00000122 7F0C                jg check_B ; если 'A>C', то переход на м
30 00000124 8B0D[39000000]    mov ecx,[C] ; иначе 'ecx = C'
31 0000012A 890D[00000000]    mov [max],ecx ; 'max = C'
32                               ; ----- Преобразование 'max(A,C)' и
33                               check_B:
34 00000130 B8[00000000]    mov eax,max
35 00000135 E862FFFFFF        call atoi ; Вызов подпрограммы перевода
36 0000013A A3[00000000]    mov [max],eax ; запись преобразованного
37                               ; ----- Сравниваем 'max(A,C)' и 'B'
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 н
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 E8ACFEFFFF        call sprint ; Вывод сообщения 'Наибольше
47 00000163 A1[00000000]    mov eax,[max]
```

Рис. 2.11: Файл листинга

11. Удаляем в файле программы один из двух операндов инструкции. Пробуем выполнить трансляцию, но получаем ошибку.

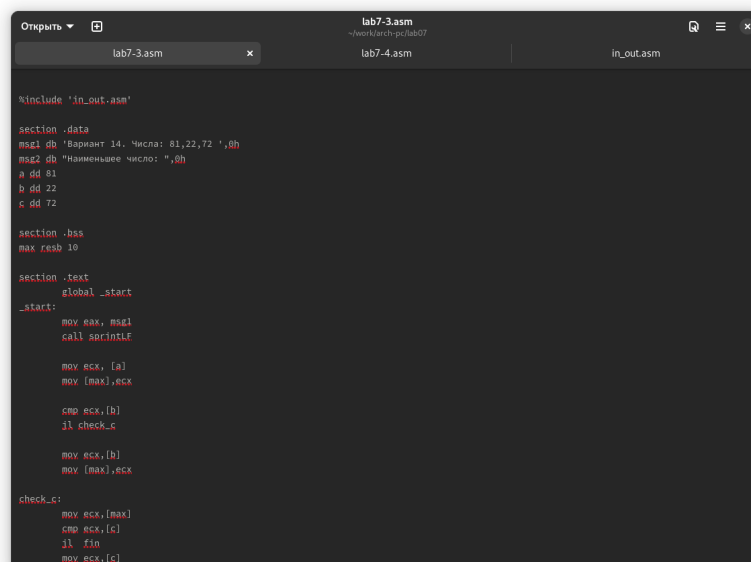


```
yaroslav@fedora: ~/work/arch-pc/lab07
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
yaroslav@fedora ~/work/arch-pc/lab07$ mcedit lab7-2.lst
yaroslav@fedora ~/work/arch-pc/lab07$ mcedit lab7-2.lst
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
yaroslav@fedora ~/work/arch-pc/lab07$ mcedit lab7-2.lst
yaroslav@fedora ~/work/arch-pc/lab07$
```

Рис. 2.12: Ошибка

3 Выполнение самостоятельной работы

1. Создаём файл lab7-3.asm и пишем в нём текст программы (14 вариант).



```
Открыть ▾ lab7-3.asm ~\work\arch-ipc\lab07
lab7-3.asm lab7-4.asm in_out.asm

#include 'in_out.asm'

section .data
msg1 db 'Вариант 14. Числа: 81,22,72 ','0h
msg2 db "Наименьшее число: ",0h
a dd 81
b dd 22
c dd 72

section .bss
max resb 10

section .text
global _start
_start:
    mov eax, msg1
    call sprintf
    mov ecx, [a]
    mov [max],ecx

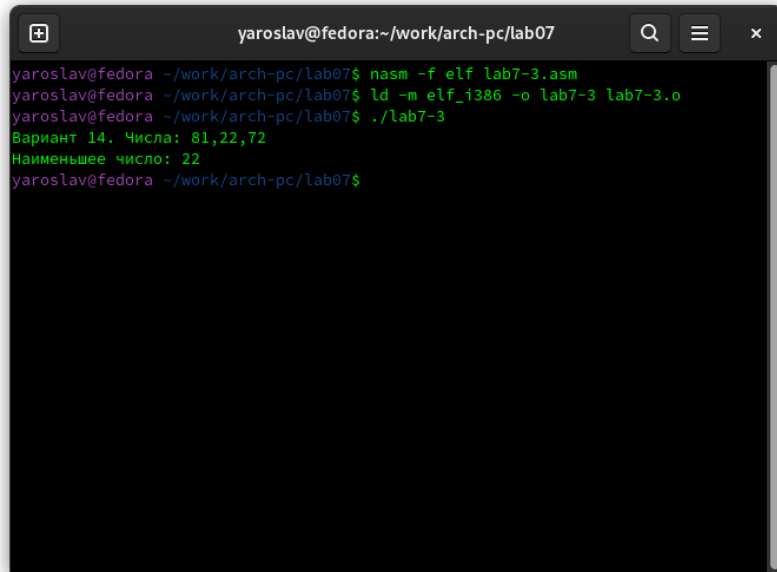
    cmp ecx,[b]
    jl check_c

    mov ecx,[b]
    mov [max],ecx

check_c:
    mov ecx,[max]
    cmp ecx,[c]
    jl fin
    mov ecx,[c]
```

Рис. 3.1: Готовый lab7-3.asm

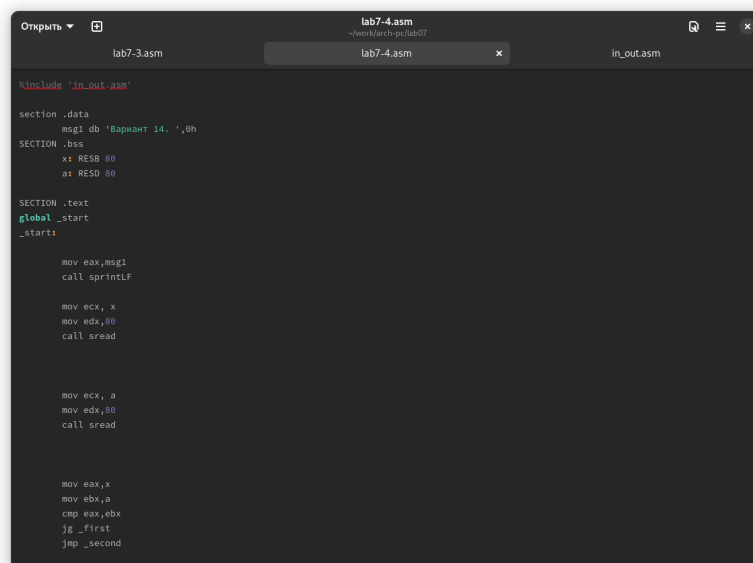
2. Проверяем работу.



```
yaroslav@fedora: ~/work/arch-pc/lab07
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
yaroslav@fedora ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-3 lab7-3.o
yaroslav@fedora ~/work/arch-pc/lab07$ ./lab7-3
Вариант 14. Числа: 81,22,72
Наименьшее число: 22
yaroslav@fedora ~/work/arch-pc/lab07$
```

Рис. 3.2: Работа программы

3. Создаём файл lab7-4.asm и пишем в нём текст программы (14 вариант).



```
lab7-4.asm
~work/arch-pc/lab07

#include "in_out.asm"

section .data
    msg1 db "Вариант 14. ",0h
SECTION .bss
    x: RESB 80
    at: RESD 80

SECTION .text
global _start
_start:

    mov eax,msg1
    call sprintf

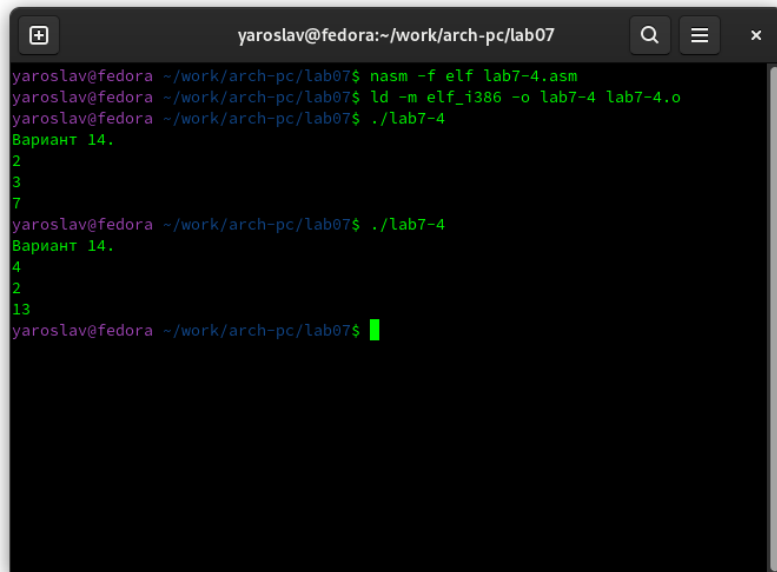
    mov ecx,x
    mov edx,80
    call sread

    mov ecx,a
    mov edx,80
    call sread

    mov eax,x
    mov ebx,a
    cmp eax,ebx
    jg _first
    jmp _second
```

Рис. 3.3: Готовый lab7-4.asm

4. Проверяем работу на заданных числах.



```
yaroslav@fedora:~/work/arch-pc/lab07
yaroslav@fedora ~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm
yaroslav@fedora ~/work/arch-pc/lab07$ ld -m elf_i386 -o lab7-4 lab7-4.o
yaroslav@fedora ~/work/arch-pc/lab07$ ./lab7-4
Вариант 14.
2
3
7
yaroslav@fedora ~/work/arch-pc/lab07$ ./lab7-4
Вариант 14.
4
2
13
yaroslav@fedora ~/work/arch-pc/lab07$
```

Рис. 3.4: Работа программы

4 Выводы

Были изучены условные и безусловные переходы, а также структура и назначение файлов листинга.