

Отчёт по лабораторной работе 7

Архитектура компьютеров

ТРАОРЕ АНРИ НОЭЛЬ

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Самостоятельное задание	16
3	Выводы	20

Список иллюстраций

2.1	Программа в файле lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	7
2.3	Программа в файле lab7-1.asm	8
2.4	Запуск программы lab7-1.asm	9
2.5	Программа в файле lab7-1.asm	10
2.6	Запуск программы lab7-1.asm	10
2.7	Программа в файле lab7-2.asm	12
2.8	Запуск программы lab7-2.asm	13
2.9	Файл листинга lab7-2	13
2.10	Ошибка трансляции lab7-2	15
2.11	Файл листинга с ошибкой lab7-2	15
2.12	Программа в файле lab7-3.asm	16
2.13	Запуск программы lab7-3.asm	17
2.14	Программа в файле lab7-4.asm	18
2.15	Запуск программы lab7-4.asm	19

Список таблиц

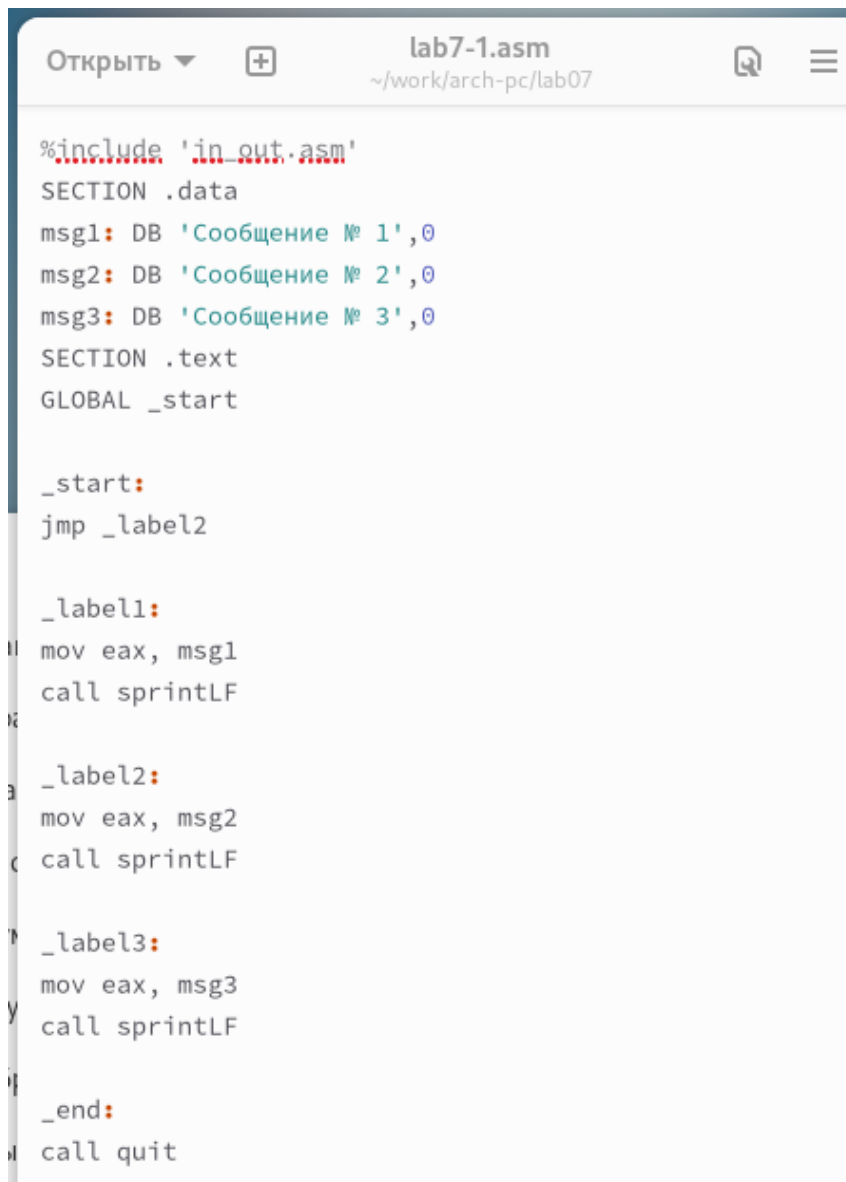
1 Цель работы

Целью 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

Создал каталог для программ лабораторной работы № 7 и файл lab7-1.asm.

Инструкция jmp в NASM используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции jmp. Написал в файл lab7-1.asm текст программы из листинга 7.1. (рис. 2.1)



```
Открыть ▾ + lab7-1.asm ~/work/arch-pc/lab07

%include 'in_out.asm'
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
call sprintLF

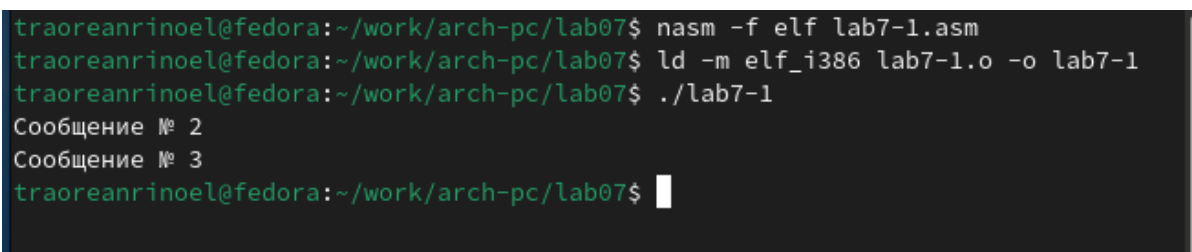
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.1: Программа в файле lab7-1.asm

Создал исполняемый файл и запустил его. (рис. 2.2)

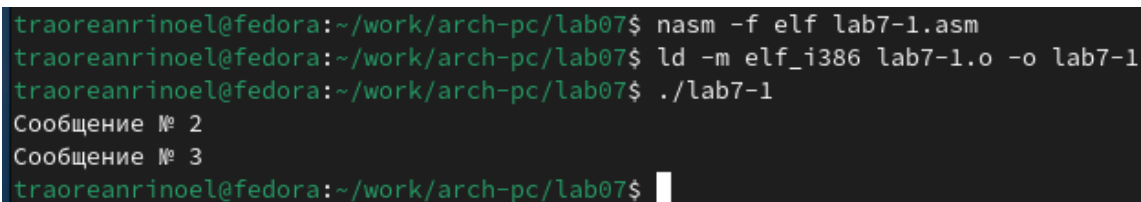


```
traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
traoreanrinoel@fedora:~/work/arch-pc/lab07$
```

Рис. 2.2: Запуск программы lab7-1.asm

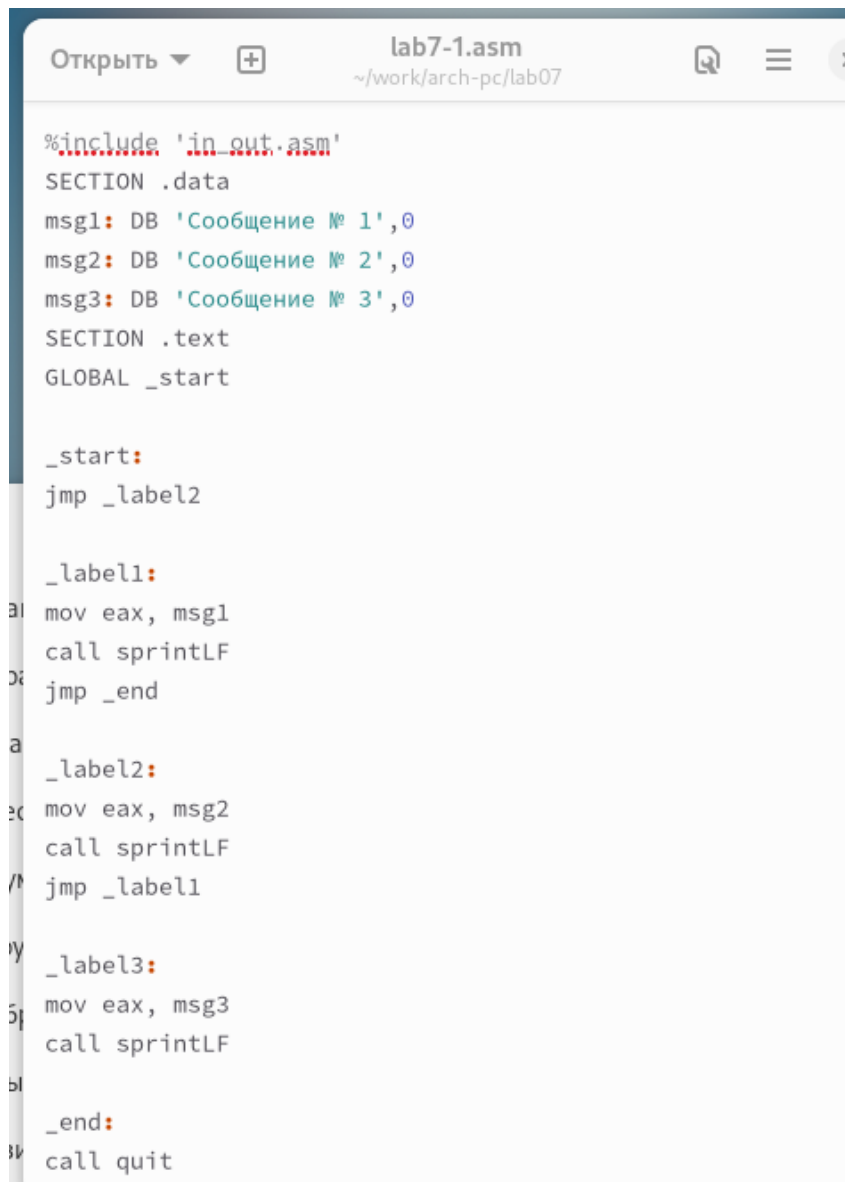
Инструкция `jmp` позволяет осуществлять переходы не только вперед но и назад. Изменим программу таким образом, чтобы она выводила сначала ‘Сообщение № 2’, потом ‘Сообщение № 1’ и завершала работу. Для этого в текст программы после вывода сообщения № 2 добавим инструкцию `jmp` с меткой `_label1` (т.е. переход к инструкциям вывода сообщения № 1) и после вывода сообщения № 1 добавим инструкцию `jmp` с меткой `_end` (т.е. переход к инструкции `call quit`).

Изменил текст программы в соответствии с листингом 7.2. (рис. 2.3) (рис. 2.4)



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

Рис. 2.3: Программа в файле lab7-1.asm



```
Открыть ▾ + lab7-1.asm  
~/work/arch-pc/lab07  
%include 'in_out.asm'  
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  
call sprintLF  
jmp _end  
  
_label2:  
mov eax, msg2  
call sprintLF  
jmp _label1  
  
_label3:  
mov eax, msg3  
call sprintLF  
  
_end:  
call quit
```

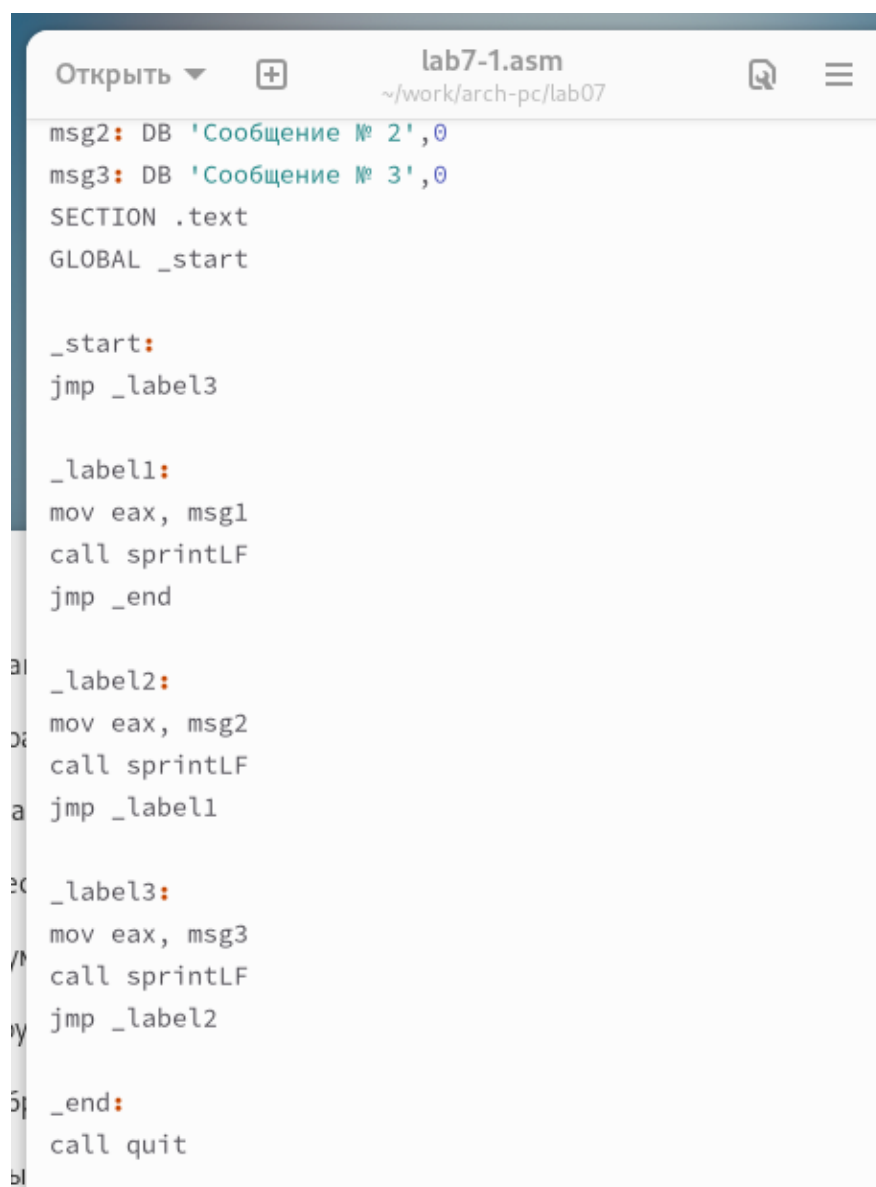
Рис. 2.4: Запуск программы lab7-1.asm

Изменил текст программы, изменив инструкции jmp, чтобы вывод программы был следующим (рис. 2.5) (рис. 2.6):

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
Открыть ▾ + lab7-1.asm ~/\work/arch-pc/lab07
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
SECTION .text
GLOBAL _start

_start:
jmp _label3

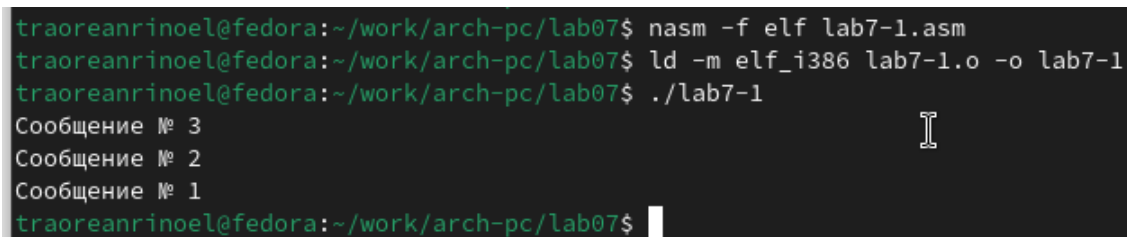
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF
jmp _label2

_end:
call quit
```

Рис. 2.5: Программа в файле lab7-1.asm

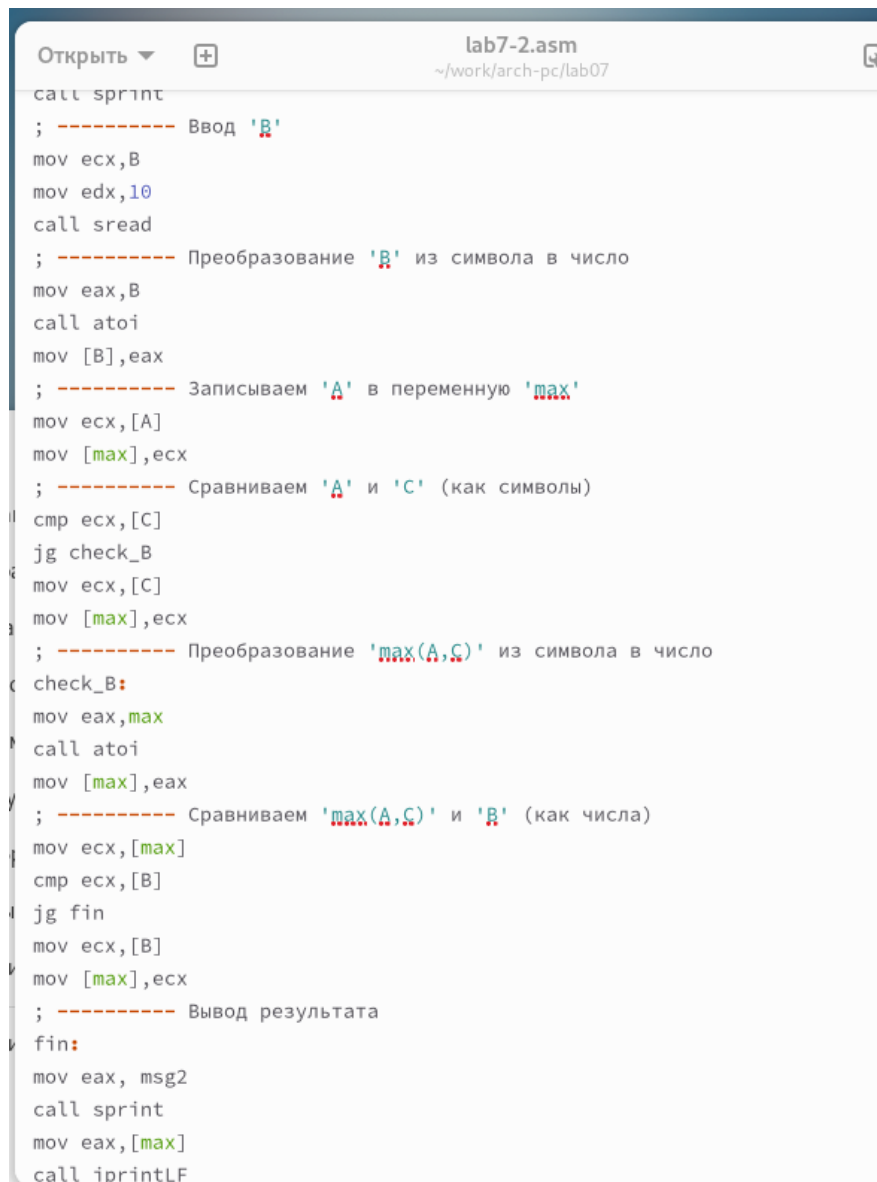


```
traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
traoreanrinoel@fedora:~/work/arch-pc/lab07$
```

Рис. 2.6: Запуск программы lab7-1.asm

Использование инструкции `jmp` приводит к переходу в любом случае. Однако, часто при написании программ необходимо использовать условные переходы, т.е. переход должен происходить если выполнено какое-либо условие. В качестве примера рассмотрим программу, 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.

Создал исполняемый файл и проверил его работу для разных значений В (рис. 2.7) (рис. 2.8).



```
Открыть ▾ + lab7-2.asm
~/work/arch-pc/lab07

call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
call sread
; ----- Преобразование 'B' из символа в число
mov eax,B
call atoi
mov [B],eax
; ----- Записываем 'A' в переменную 'max'
mov ecx,[A]
mov [max],ecx
; ----- Сравниваем 'A' и 'C' (как символы)
cmp ecx,[C]
jg check_B
mov ecx,[C]
mov [max],ecx
; ----- Преобразование 'max(A,C)' из символа в число
check_B:
mov eax,max
call atoi
mov [max],eax
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B]
jg fin
mov ecx,[B]
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg2
call sprint
mov eax,[max]
call iprintLF
```

Рис. 2.7: Программа в файле lab7-2.asm

```

traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 20
Наибольшее число: 50
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 40
Наибольшее число: 50
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
traoreanrinoel@fedora:~/work/arch-pc/lab07$

```

Рис. 2.8: Запуск программы lab7-2.asm

Обычно nasm создаёт в результате ассемблирования только объектный файл. Получить файл листинга можно, указав ключ -l и задав имя файла листинга в командной строке.

Создал файл листинга для программы из файла lab7-2.asm (рис. 2.9)

```

10      section .text
11      global _start
12      _start:
13      ; ----- Вывод сообщения 'Введите B: '
14      000000f8 B8[00000000] mov eax,msg1
15      000000fd EB10FFFFFF call printf
16      ; ----- Ввод 'B'
17      000000f2 B9[0A000000] mov ecx,B
18      000000f7 BA0A000000 mov edx,10
19      000000fc EA42FFFFFF call sread
20      ; ----- Преобразование 'B' из символа в число
21      00000101 B8[0A000000] mov eax,B
22      00000106 EB91FFFFFF call atoi
23      0000010B A3[0A000000] mov [B],eax
24      ; ----- Записываем 'A' в переменную 'max'
25      00000110 8B0D[35000000] mov ecx,[A]
26      00000116 890D[00000000] mov [max],ecx
27      ; ----- Сравниваем 'A' и 'C' (как символы)
28      0000011C 3B0D[39000000] cmp ecx,[C]
29      00000122 7F0C      jg check_B
30      00000124 8B0D[39000000] mov ecx,[C]
31      0000012A 890D[00000000] mov [max],ecx
32      ; ----- Преобразование 'max(A,C)' из символа в число
33      check_B:
34      00000130 B8[00000000] mov eax,max
35      00000135 EB62FFFFFF call atoi
36      0000013A A3[00000000] mov [max],eax
37      ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38      0000013F 8B0D[00000000] mov ecx,[max]
39      00000145 3B0D[0A000000] cmp ecx,[B]
40      0000014B 7F0C      jg .fin
41      0000014D 8B0D[0A000000] mov ecx,[B]

```

Рис. 2.9: Файл листинга lab7-2

Внимательно ознакомился с его форматом и содержимым. Подробно объясню содержимое трёх строк файла листинга по выбору.

строка 189

- 14 - номер строки в подпрограмме
- 000000E8 - адрес
- B8[00000000] - машинный код
- mov eax,msg1 - код программы - перекладывает msg1 в eax

строка 190

- 15 - номер строки в подпрограмме
- 000000ED - адрес
- E81DFFFFFF - машинный код
- call sprint - код программы - вызов подпрограммы печати

строка 192

- 17 - номер строки в подпрограмме
- 000000F2 - адрес
- B9[0A000000] - машинный код
- mov ecx,B - код программы - перекладывает B в ecx

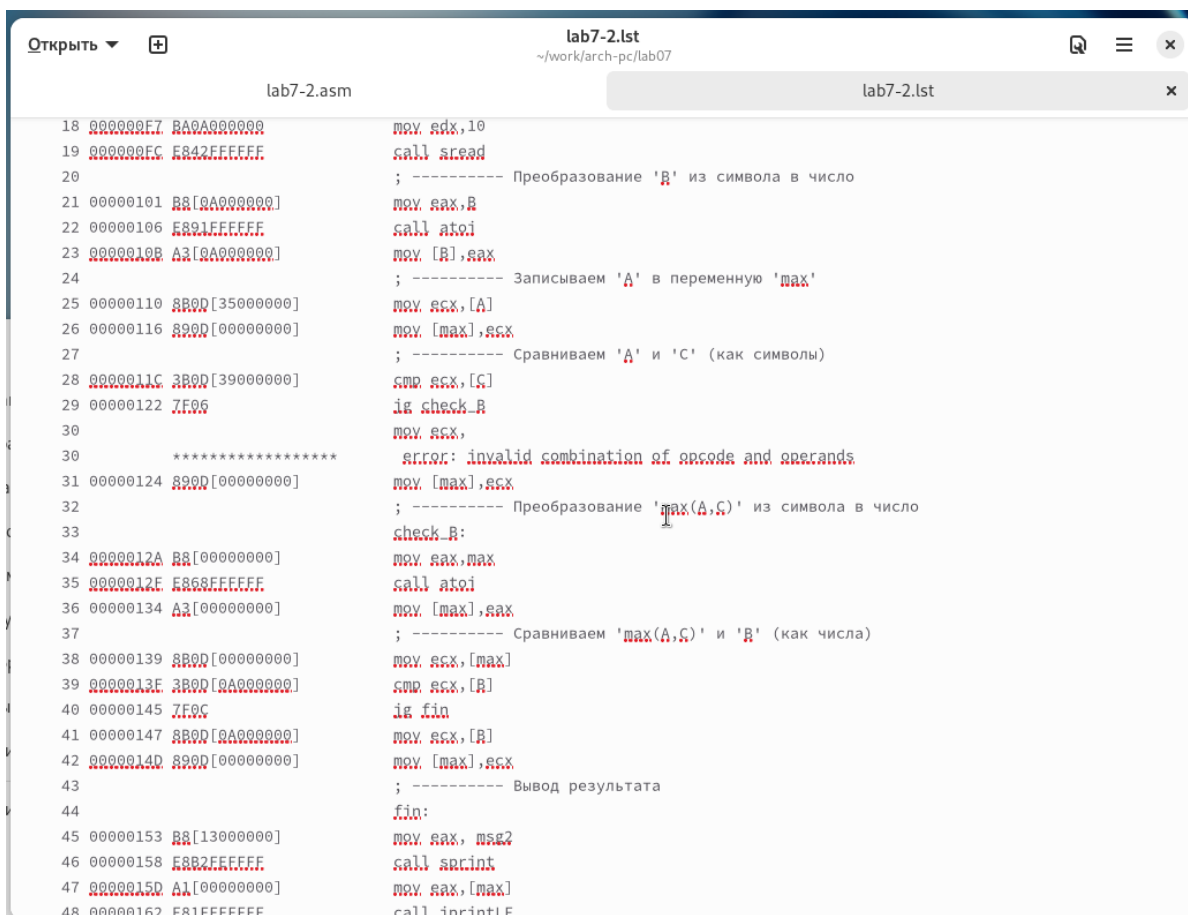
Открыл файл с программой lab7-2.asm и в инструкции с двумя операндами удалил один операнд. Выполнил трансляцию с получением файла листинга. (рис. 2.10) (рис. 2.11)

```

traoreanrinoel@fedora:~/work/arch-pc/lab07$
traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:30: error: invalid combination of opcode and operands
traoreanrinoel@fedora:~/work/arch-pc/lab07$

```

Рис. 2.10: Ошибка трансляции lab7-2



```

lab7-2.asm
18 000000f7 BA0A000000 mov edx,10
19 000000fc E842FFFFFF call sread
20 ; ----- Преобразование 'B' из символа в число
21 00000101 8B[0A000000] mov eax,B
22 00000106 E801FFFFFF call atoi
23 0000010B A3[0A000000] mov [B],eax
24 ; ----- Записываем 'A' в переменную 'max'
25 00000110 8B0D[35000000] mov ecx,[A]
26 00000116 890D[00000000] mov [max],ecx
27 ; ----- Сравниваем 'A' и 'C' (как символы)
28 0000011C 3B0D[39000000] cmp ecx,[C]
29 00000122 7F0C jg check_B
30 ***** error: invalid combination of opcode and operands
31 00000124 890D[00000000] mov [max],ecx
32 ; ----- Преобразование 'max(A,C)' из символа в число
33 check_B:
34 0000012A 8B[00000000] mov eax,max
35 0000012E E808FFFFFF call atoi
36 00000134 A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 00000139 8B0D[00000000] mov ecx,[max]
39 0000013F 3B0D[0A000000] cmp ecx,[B]
40 00000145 7F0C jg fin
41 00000147 8B0D[0A000000] mov ecx,[B]
42 0000014D 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000153 8B[13000000] mov eax,msg2
46 00000158 E8B2FFFFFF call sprintf
47 0000015D A3[00000000] mov eax,[max]
48 00000162 E81FFFFFFF call idprintf

```

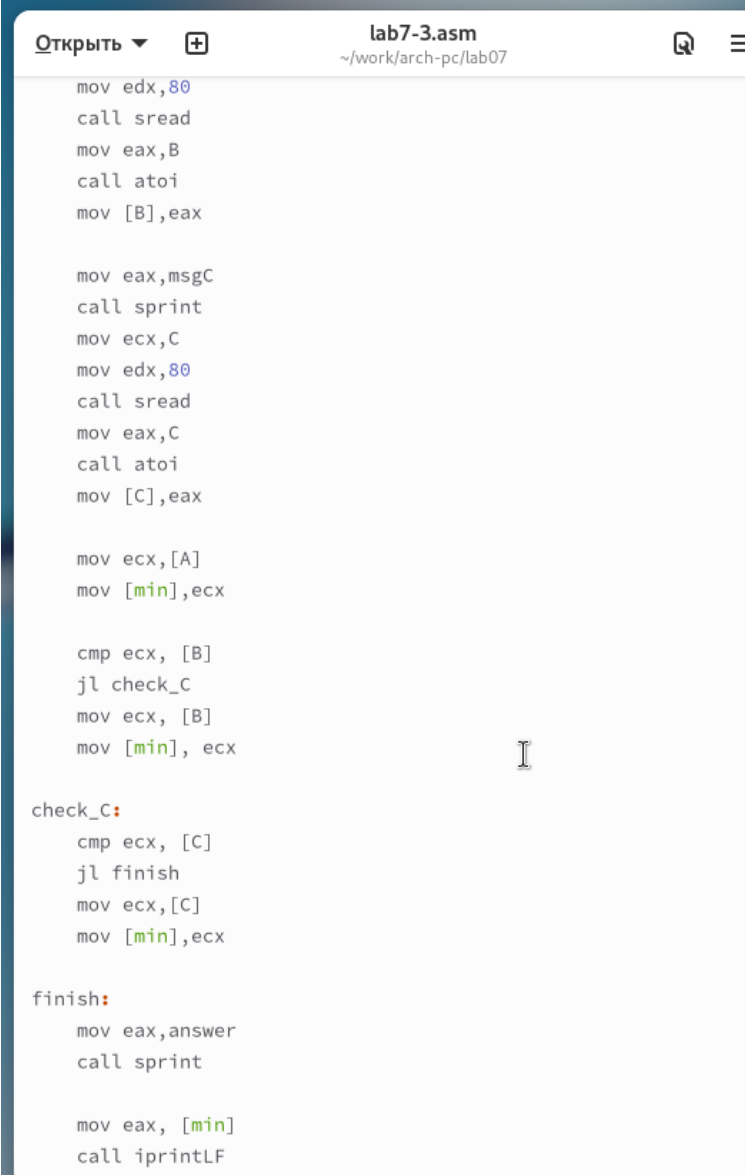
Рис. 2.11: Файл листинга с ошибкой lab7-2

Объектный файл не смог создаться из-за ошибки. Но получился листинг, где выделено место ошибки.

2.1 Самостоятельное задание

Напишите программу нахождения наименьшей из 3 целочисленных переменных a, b и c. Значения переменных выбрать из табл. 7.5 в соответствии с вариантом, полученным при выполнении лабораторной работы № 6. Создайте исполняемый файл и проверьте его работу (рис. 2.12) (рис. 2.13)

для варианта 19 - 46,32,74



```
lab7-3.asm
~/work/arch-pc/lab07

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

mov eax,msgC
call sprint
mov ecx,C
mov edx,80
call sread
mov eax,C
call atoi
mov [C],eax

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

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

check_C:
    cmp ecx, [C]
    jl finish
    mov ecx,[C]
    mov [min],ecx

finish:
    mov eax,answer
    call sprint

    mov eax, [min]
    call iprintLF
```

Рис. 2.12: Программа в файле lab7-3.asm


```

traoreanrinoel@fedora:~/work/arch-pc/lab07$
traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-3.asm
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-3.o -o lab7-3
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-3
Input A: 46
Input B: 32
Input C: 74
Smallest: 32
traoreanrinoel@fedora:~/work/arch-pc/lab07$ █

```

Рис. 2.13: Запуск программы lab7-3.asm

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. (рис. 2.14) (рис. 2.15)

для варианта 19

$$\begin{cases} a + x, & x > a \\ x, & x \leq a \end{cases}$$

Если подставить $x = 4, a = 5$ получается 4.

Если подставить $x = 3, a = 2$ получается 5.

```
mov eax,msgA
call sprint
mov ecx,A
mov edx,80
call sread
mov eax,A
call atoi
mov [A],eax

mov eax,msgX
call sprint
mov ecx,X
mov edx,80
call sread
mov eax,X
call atoi
mov [X],eax

mov ebx, [X]
mov edx, [A]
cmp ebx, edx
ja first
jmp second

first:
mov eax,[A]
add eax,[X]
call iprintLF
call quit

second:
mov eax,[X]
call iprintLF
call quit
```

Рис. 2.14: Программа в файле lab7-4.asm

```
traoreanrinoel@fedora:~/work/arch-pc/lab07$  
traoreanrinoel@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-4.asm  
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-4.o -o lab7-4  
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 5  
Input X: 4  
4  
traoreanrinoel@fedora:~/work/arch-pc/lab07$ ./lab7-4  
Input A: 2  
Input X: 3  
5  
traoreanrinoel@fedora:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск программы lab7-4.asm

3 Выводы

Изучили команды условного и безусловного переходов, познакомились с фалом листинга.