

# **Отчёт по лабораторной работе 7**

**дисциплина: Архитектура компьютеров**

Грачев Я. М. НПИбд-01-24

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
2.1	Реализация переходов в NASM . . . . .	6
2.2	Изучение структуры файла листинга . . . . .	12
2.3	Самостоятельное задание . . . . .	15
<b>3</b>	<b>Выводы</b>	<b>20</b>

## Список иллюстраций

2.1	Создан каталог . . . . .	6
2.2	Программа lab7-1.asm . . . . .	7
2.3	Запуск программы lab7-1.asm . . . . .	8
2.4	Программа lab7-1.asm . . . . .	8
2.5	Запуск программы lab7-1.asm . . . . .	9
2.6	Программа lab7-1.asm . . . . .	10
2.7	Запуск программы lab7-1.asm . . . . .	10
2.8	Программа lab7-2.asm . . . . .	11
2.9	Запуск программы lab7-2.asm . . . . .	12
2.10	Файл листинга lab7-2 . . . . .	13
2.11	Ошибка трансляции lab7-2 . . . . .	14
2.12	Файл листинга с ошибкой lab7-2 . . . . .	15
2.13	Программа lab7-task1.asm . . . . .	16
2.14	Запуск программы lab7-task1.asm . . . . .	17
2.15	Программа lab7-task2.asm . . . . .	18
2.16	Запуск программы lab7-task2.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

### 2.1 Реализация переходов в NASM

Создал каталог для программ лабораторной работы № 7 и файл `lab7-1.asm` (рис. 2.1).

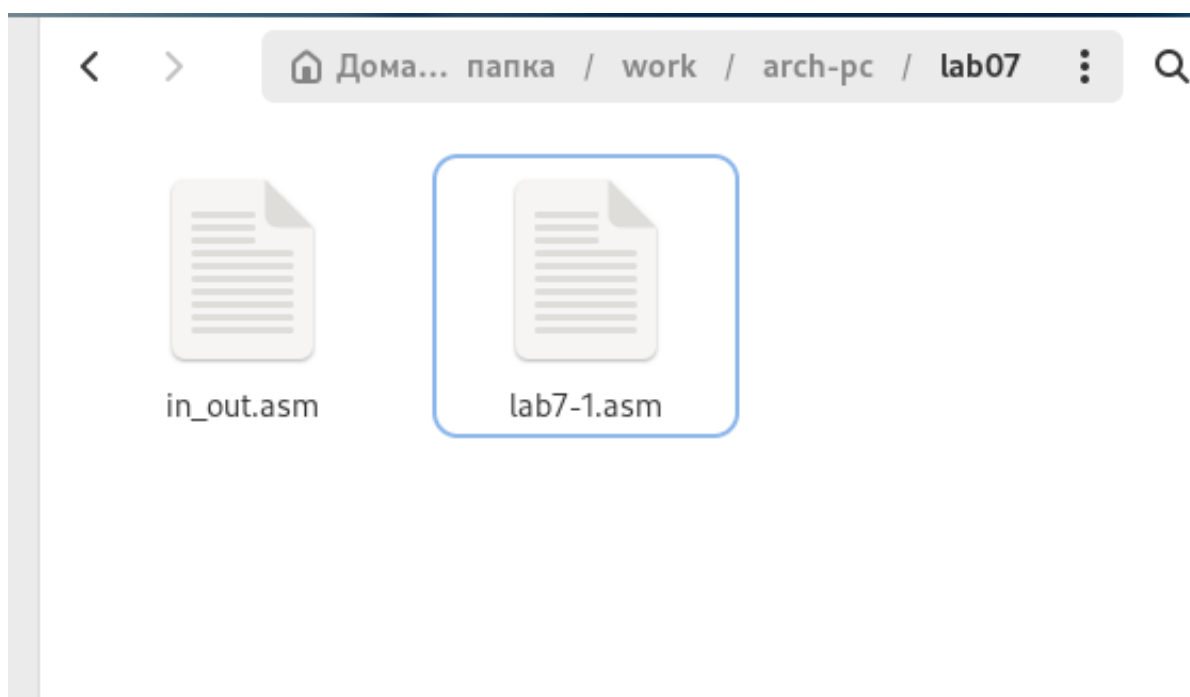
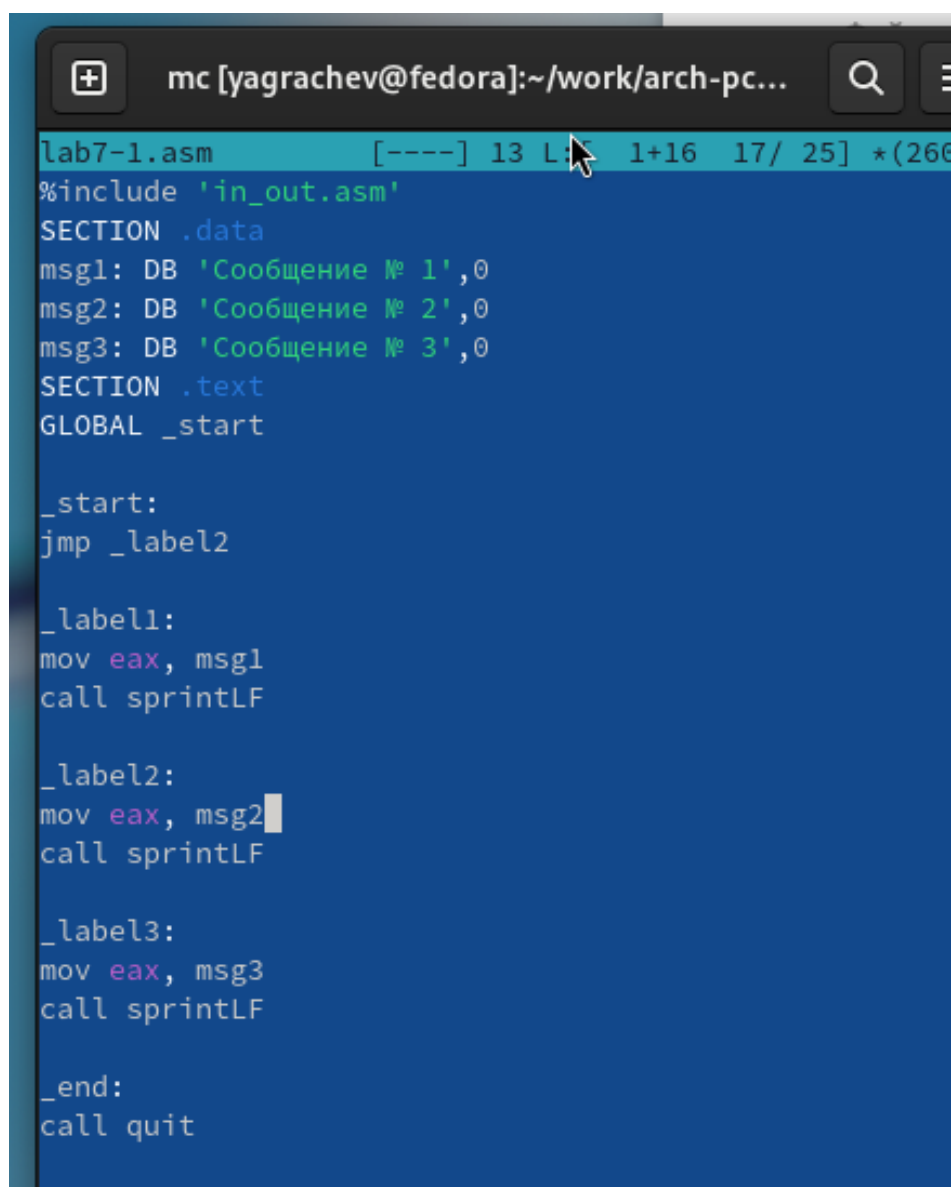


Рис. 2.1: Создан каталог

В NASM инструкция `jmp` используется для реализации безусловных переходов. Рассмотрим пример программы с использованием инструкции `jmp`. В файле `lab7-1.asm` разместил текст программы из листинга 7.1 (рис. 2.2).



```
mc [yagrachev@fedora]:~/work/arch-pc...
lab7-1.asm [----] 13 L: 1+16 17/ 25] *(260
%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. 2.2: Программа lab7-1.asm

Создал исполняемый файл и запустил его (рис. 2.3).

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

Рис. 2.3: Запуск программы lab7-1.asm

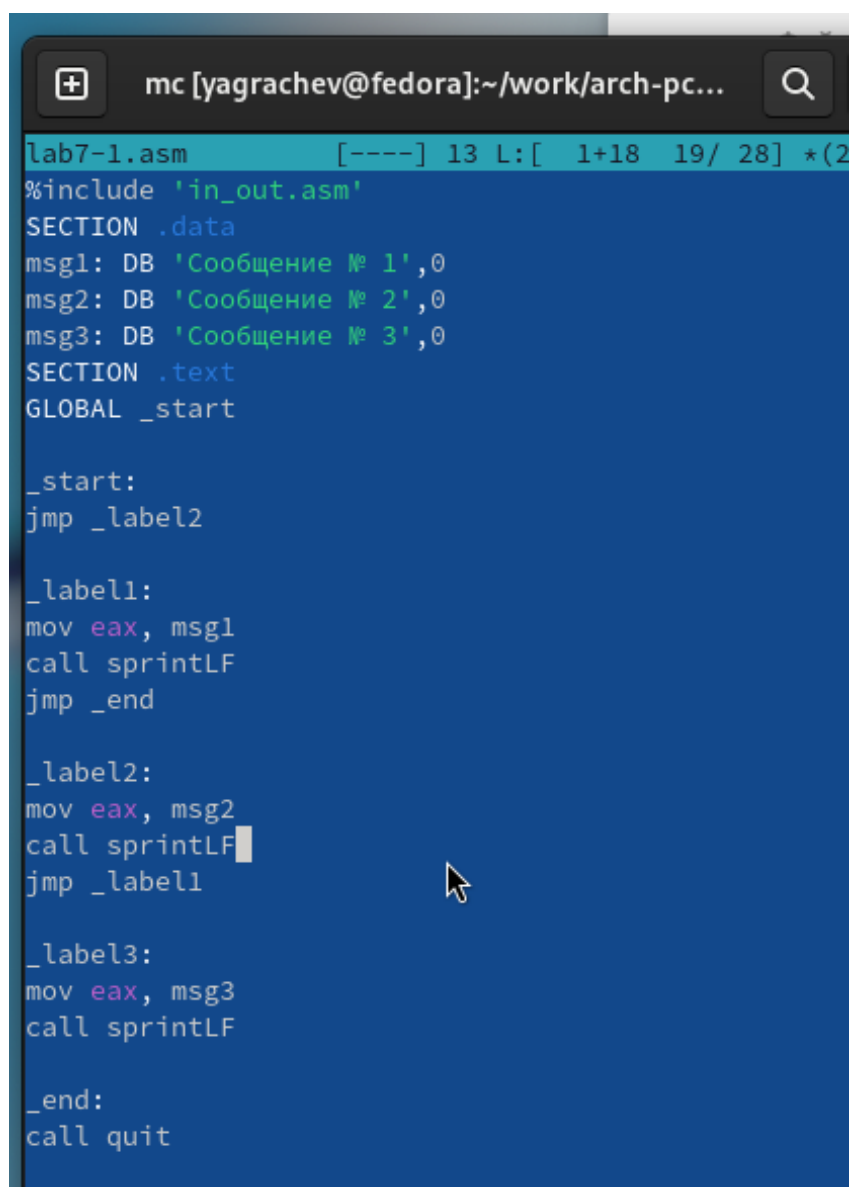
Инструкция `jmp` позволяет выполнять переходы как вперёд, так и назад. Изменил программу так, чтобы сначала выводилось сообщение № 2, затем сообщение № 1, после чего программа завершала работу. Для этого добавил в текст программы инструкцию `jmp` с меткой `_label1` после вывода сообщения № 2 (чтобы перейти к инструкции вывода сообщения № 1) и инструкцию `jmp` с меткой `_end` после вывода сообщения № 1 (для перехода к инструкции `call quit`).

Обновил текст программы согласно листингу 7.2 (рис. 2.4 и 2.5).

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

Рис. 2.4: Программа lab7-1.asm





```
lab7-1.asm [----] 13 L: [ 1+18 19/ 28] *(2
#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. 2.5: Запуск программы lab7-1.asm

Изменил текст программы так, чтобы итоговый вывод программы выглядел следующим образом (рис. 2.6 и 2.7):

Сообщение № 3 Сообщение № 2 Сообщение № 1

```

lab7-1.asm      [----]  8 L:[  1+16  17/ 29] *(255 [*] [
#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 _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. 2.6: Программа lab7-1.asm

```

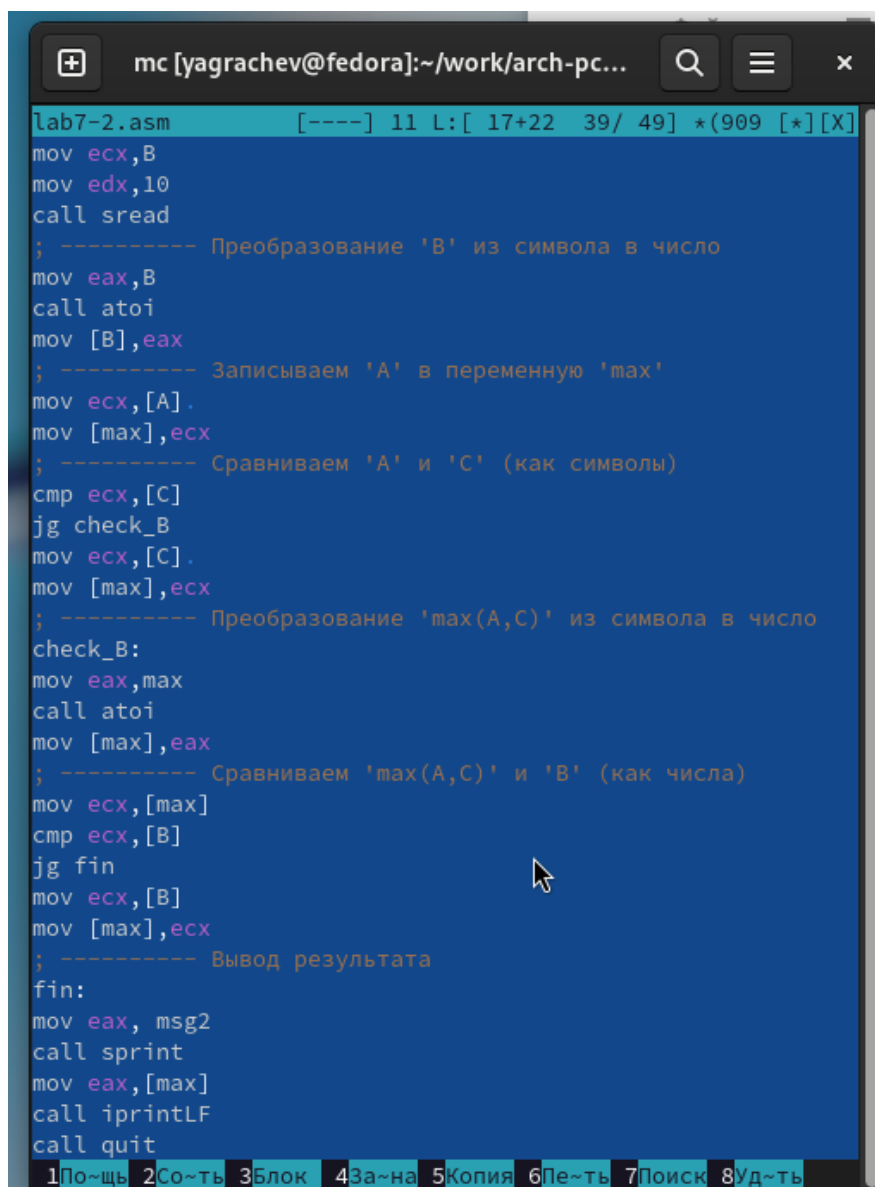
yagrachev@fedora:~/work/arch-pc/lab07$
yagrachev@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-1.asm
yagrachev@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
yagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
yagrachev@fedora:~/work/arch-pc/lab07$

```

Рис. 2.7: Запуск программы lab7-1.asm

Инструкция `jmp` всегда вызывает переход. Однако часто в программировании требуются условные переходы, которые выполняются только при соблюдении определённых условий. В качестве примера рассмотрим программу, определяющую и выводящую наибольшее значение среди трёх целочисленных переменных A, B и C. Значения для A и C заданы в программе, а B вводится с клавиатуры.

Создал исполняемый файл и проверил его работу для различных значений B (рис. 2.8 и 2.9).



```
lab7-2.asm      [----] 11 L: [ 17+22 39/ 49] *(909 [*] [X])
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
call quit
1По~щъ 2Со~тъ 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть
```

Рис. 2.8: Программа lab7-2.asm

```

yaagrachev@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm
yaagrachev@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
yaagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 17
Наибольшее число: 50
yaagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 55
Наибольшее число: 55
yaagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-2
Введите В: 70
Наибольшее число: 70
yaagrachev@fedora:~/work/arch-pc/lab07$

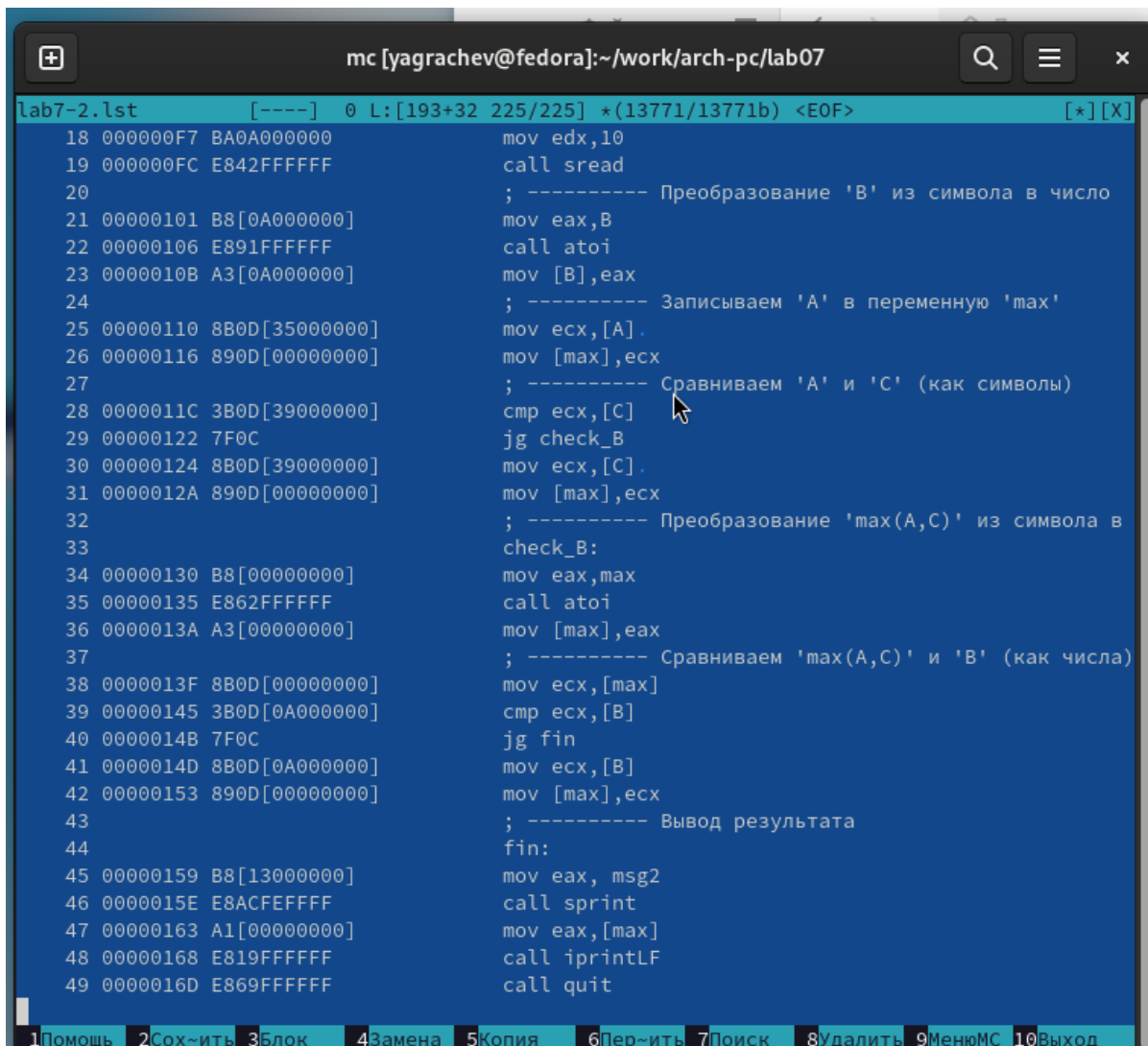
```

Рис. 2.9: Запуск программы lab7-2.asm

## 2.2 Изучение структуры файла листинга

Обычно NASM создаёт только объектный файл. Чтобы получить файл листинга, нужно указать ключ `-l` и задать имя файла листинга в командной строке.

Создал файл листинга для программы из файла `lab7-2.asm` (рис. 2.10).



```
lab7-2.lst [----] 0 L:[193+32 225/225] *(13771/13771b) <EOF> [*] [X]
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
22 00000106 E891FFFFFF 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
33 check_B:
34 00000130 B8[00000000] mov eax,max
35 00000135 E862FFFFFF 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]
42 00000153 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000159 B8[13000000] mov eax,msg2
46 0000015E E8ACFEFFFF call sprint
47 00000163 A1[00000000] mov eax,[max]
48 00000168 E819FFFFFF call iprintLF
49 0000016D E869FFFFFF call quit
```

Рис. 2.10: Файл листинга lab7-2

Рассмотрим его структуру:

- **Строка 211**

- 34 — номер строки
- 0000012E — адрес
- B8[00000000] — машинный код
- mov eax, max — код программы

- **Строка 212**

- 35 — номер строки
- 00000133 — адрес
- E864FFFFFF — машинный код
- `call atoi` — код программы

• **Строка 213**

- 36 — номер строки
- 00000138 — адрес
- `A3[00000000]` — машинный код
- `mov [max], eax` — код программы

Открыл файл `lab7-2.asm`, удалил один из операндов в инструкции с двумя операндами и выполнил трансляцию с получением файла листинга (рис. 2.11 и 2.12).

```
yagrachev@fedora:~/work/arch-pc/lab07$
yagrachev@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
yagrachev@fedora:~/work/arch-pc/lab07$
yagrachev@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst
lab7-2.asm:41: error: invalid combination of opcode and operands
yagrachev@fedora:~/work/arch-pc/lab07$
yagrachev@fedora:~/work/arch-pc/lab07$
```

Рис. 2.11: Ошибка трансляции lab7-2

```
lab7-2.lst      [----] 23 L:[194+25 219/226] *(13488/13859b) 0032 0x020      [*] [X]
19 000000FC E842FFFFFF      call sread
20                          ; ----- Преобразование 'B' из символа в число
21 00000101 B8[0A000000]      mov eax,B
22 00000106 E891FFFFFF      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
33 check_B:
34 00000130 B8[00000000]      mov eax,max
35 00000135 E862FFFFFF      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 7F06      jg fin
41 0000014D 890D[00000000]      mov [max],ecx
42                          ; ----- Вывод результата
43 fin:
44 00000153 B8[13000000]      mov eax,msg2
45 00000158 E8B2FEFFFF      call sprint
46 0000015D A1[00000000]      mov eax,[max]
47 00000162 E81FFFFFFF      call iprintLF
48 00000167 E86FFFFFFF      call quit
41 *****      error: invalid combination of opcode and operands
42 0000014D 890D[00000000]      mov [max],ecx
43
44
45 00000153 B8[13000000]      mov eax,msg2
46 00000158 E8B2FEFFFF      call sprint
47 0000015D A1[00000000]      mov eax,[max]
48 00000162 E81FFFFFFF      call iprintLF
49 00000167 E86FFFFFFF      call quit
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход

Рис. 2.12: Файл листинга с ошибкой lab7-2

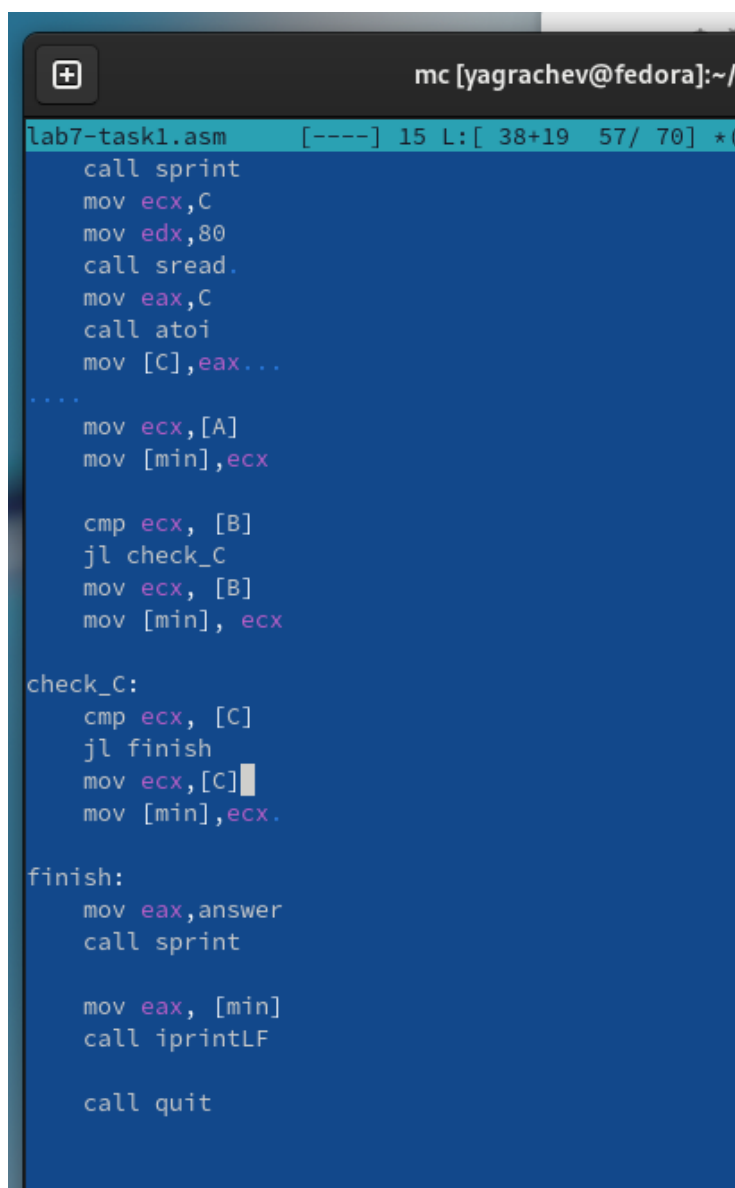
Из-за ошибки объектный файл не был создан. Однако, листинг указал место-положение ошибки.

## 2.3 Самостоятельное задание

1. Найти наименьшее среди трёх целочисленных переменных  $a$ ,  $b$  и  $c$ , используя значения из таблицы 7.5 для варианта, полученного при выполнении лабораторной работы № 6. Создать исполняемый файл и проверить

его работу (рис. 2.13 и 2.14).

Для варианта 1:  $a = 17$ ,  $b = 23$ ,  $c = 45$ .



```
lab7-task1.asm [----] 15 L: [ 38+19 57/ 70] * (
    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

    call quit
```

Рис. 2.13: Программа lab7-task1.asm



```
yagrachev@fedora:~/work/arch-pc/lab07$  
yagrachev@fedora:~/work/arch-pc/lab07$  
yagrachev@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-task1.asm  
yagrachev@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task1.o -o lab7-task1  
yagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-task1  
Input A: 17  
Input B: 23  
Input C: 45  
Smallest: 17  
yagrachev@fedora:~/work/arch-pc/lab07$
```

Рис. 2.14: Запуск программы lab7-task1.asm

2. **Программа для вычисления функции  $f(x)$**  при введённых значениях  $x$  и  $a$  с клавиатуры. Вид функции  $f(x)$  выбирается из таблицы 7.6 в зависимости от варианта, полученного для лабораторной работы № 7. Создать исполняемый файл и проверить его работу для значений  $x$  и  $a$  из таблицы 7.6 (рис. 2.15 и 2.16).

Для варианта 1:

$$f(x) = \begin{cases} 2a - x, & x < a \\ 8, & x \geq a \end{cases}$$

При  $x = 1, a = 2$  результат — 3.

При  $x = 2, a = 1$  результат — 8.

```
mc [yagrachev@fedora]:~/work/arch-pc/
lab7-task2.asm [----] 0 L: [ 19+29 48/ 51] *(656 / 690b)
    call sread
    mov  eax,A
    call atoi.
    mov  [A],eax

    mov  eax,msgX
    call sprintf
    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
    jnb  first
    jmp  second

first:
    mov  eax,[A]
    mov  ebx,2
    mul  ebx
    sub  eax,[X]
    call iprintLF.
    call quit
second:
    mov  eax,8
    call iprintLF.
    call quit
```

Рис. 2.15: Программа lab7-task2.asm

```
yagrachev@fedora:~/work/arch-pc/lab07$  
yagrachev@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-task2.asm  
yagrachev@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-task2.o -o lab7-task2  
yagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 2  
Input X: 1  
3  
yagrachev@fedora:~/work/arch-pc/lab07$ ./lab7-task2  
Input A: 1  
Input X: 2  
8  
yagrachev@fedora:~/work/arch-pc/lab07$
```

Рис. 2.16: Запуск программы lab7-task2.asm

## 3 Выводы

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