

Отчёт по лабораторной работе 7

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

Зиборова Вероника Николаевна НММбд-02-24

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Реализация переходов в NASM	6
2.2	Изучение структуры файла листинга	13
2.3	Задание для самостоятельной работы	16
3	Выводы	20

Список иллюстраций

2.1	Программа в файле lab7-1.asm	7
2.2	Запуск программы lab7-1.asm	8
2.3	Измененная программа в lab7-1.asm	9
2.4	Запуск измененной программы	9
2.5	Финальная версия программы lab7-1.asm	10
2.6	Запуск программы с новой последовательностью	11
2.7	Программа для нахождения максимума в lab7-2.asm	12
2.8	Запуск программы для нахождения максимума	13
2.9	Файл листинга lab7-2	14
2.10	Ошибка трансляции lab7-2	15
2.11	Листинг с ошибкой lab7-2	16
2.12	Программа в task7-1.asm	17
2.13	Запуск task7-1.asm	17
2.14	Программа для $f(x)$ в task7-2.asm	18
2.15	Запуск task7-2.asm	19

Список таблиц

1 Цель работы

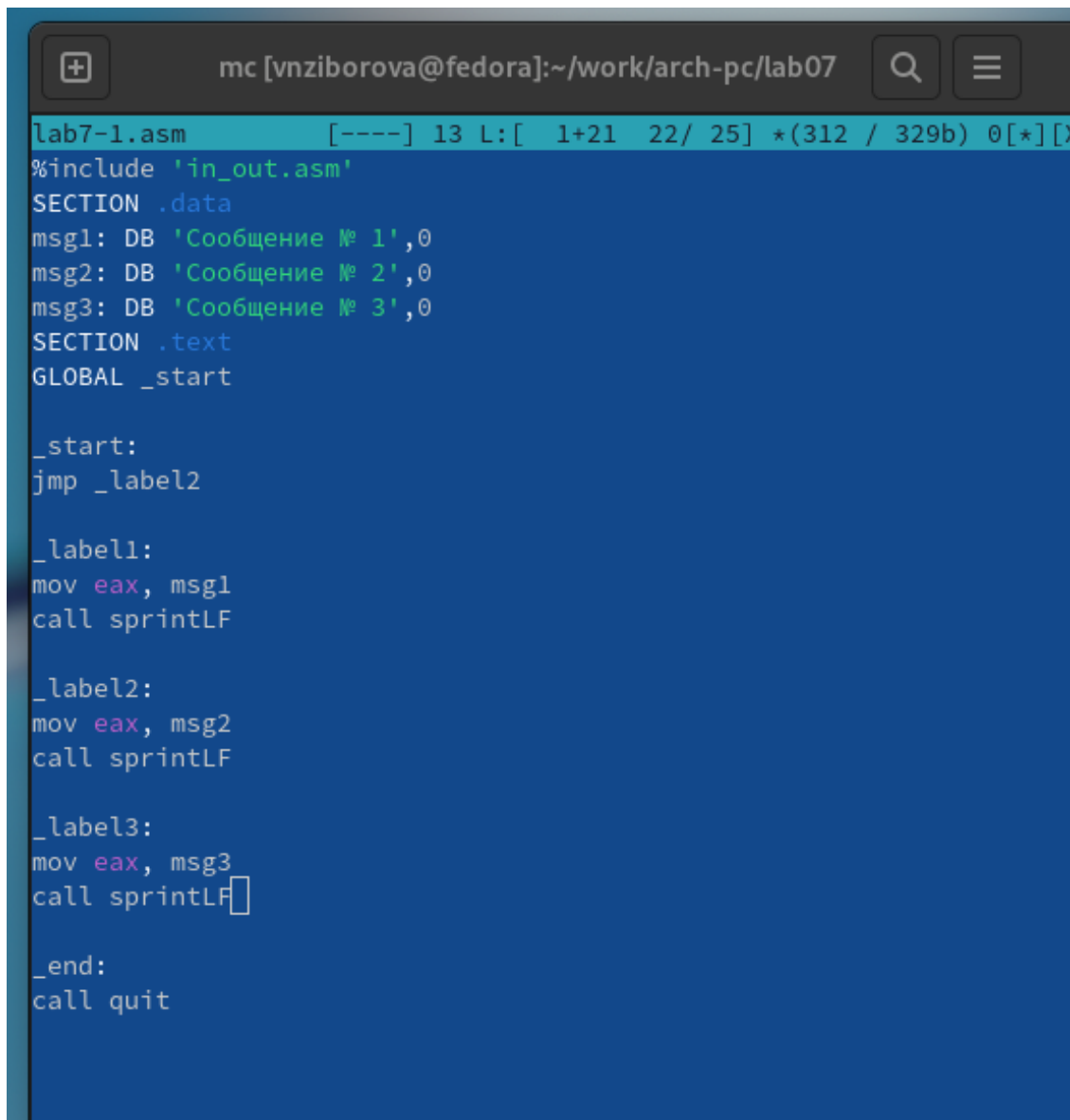
Целью работы является изучение команд условного и безусловного переходов. Приобретение навыков написания программ с использованием переходов. Знакомство с назначением и структурой файла листинга.

2 Выполнение лабораторной работы

2.1 Реализация переходов в NASM

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

В NASM инструкция jmp используется для реализации безусловных переходов. Пример программы с этой инструкцией записан в файл lab7-1.asm.



```
lab7-1.asm [----] 13 L: [ 1+21 22/ 25] *(312 / 329b) 0[*] [>
#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

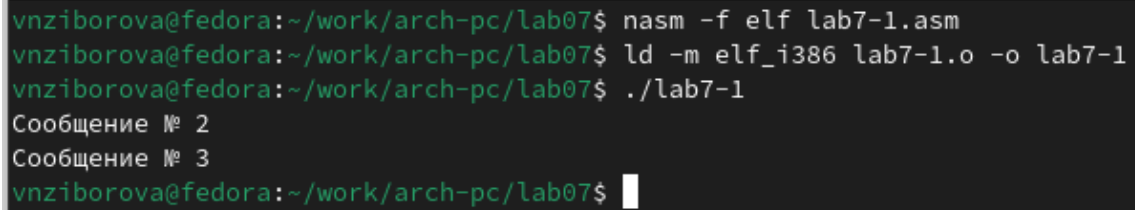
_label2:
mov eax, msg2
call sprintLF

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

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

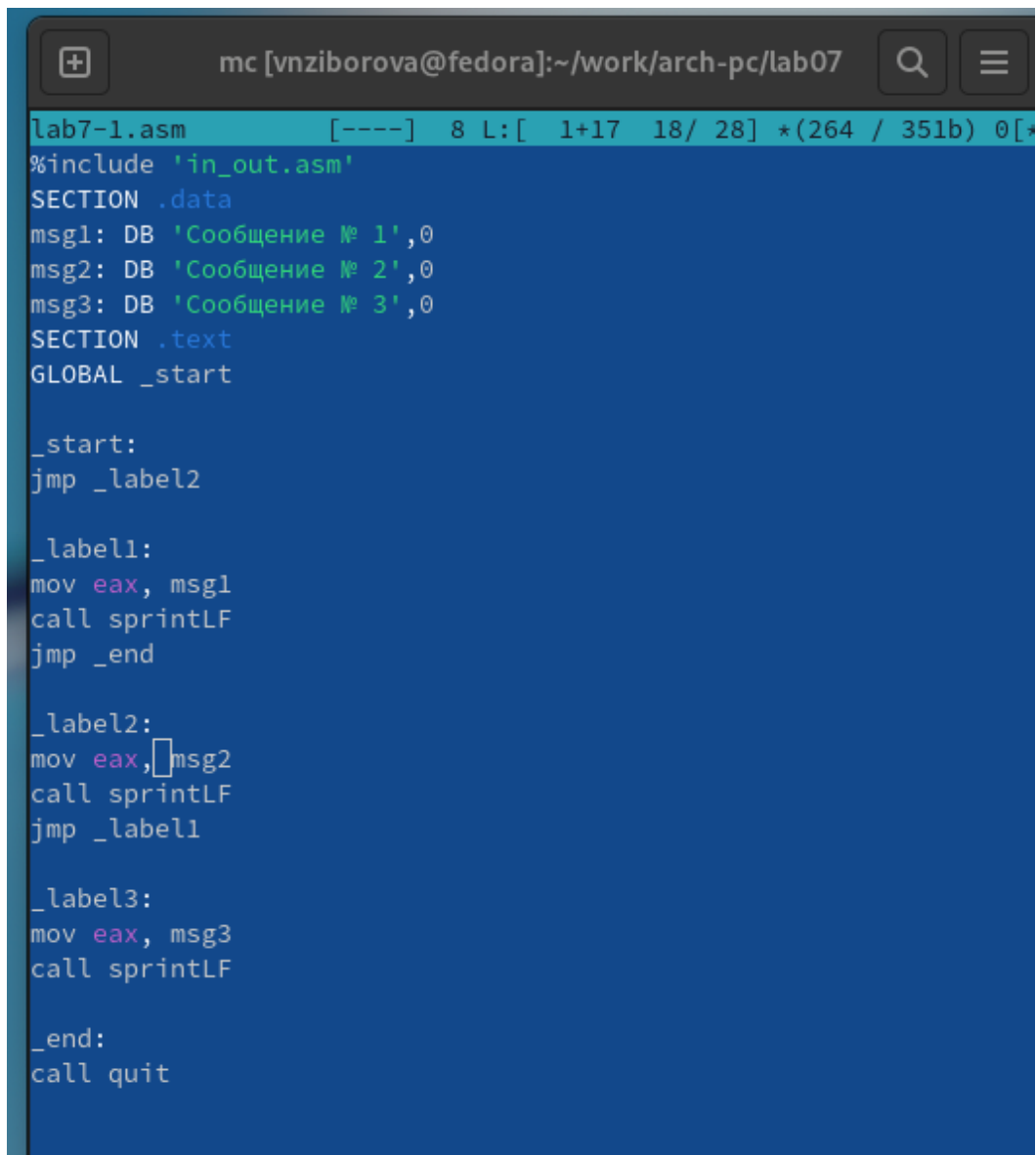
Скомпилировала программу и запустила исполняемый файл.

A terminal window with a dark background and green text. The user is at the prompt 'vnziborova@fedora:~/work/arch-pc/lab07\$'. They enter 'nasm -f elf lab7-1.asm', then 'ld -m elf_i386 lab7-1.o -o lab7-1', and finally './lab7-1'. The program outputs 'Сообщение № 2' and 'Сообщение № 3' on separate lines. The prompt returns after the execution.

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

Рис. 2.2: Запуск программы lab7-1.asm

С помощью `jmp` можно переходить как вперед, так и назад. Изменила программу так, чтобы она сначала выводила «Сообщение №2», потом «Сообщение №1» и завершала работу. Для этого добавила инструкцию `jmp` после вывода «Сообщение №2» с меткой `_label1` (переход к коду вывода «Сообщение №1») и `jmp` с меткой `_end` после «Сообщение №1» (переход к завершению).



```
lab7-1.asm [----] 8 L: [ 1+17 18/ 28] *(264 / 351b) 0[*]
#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

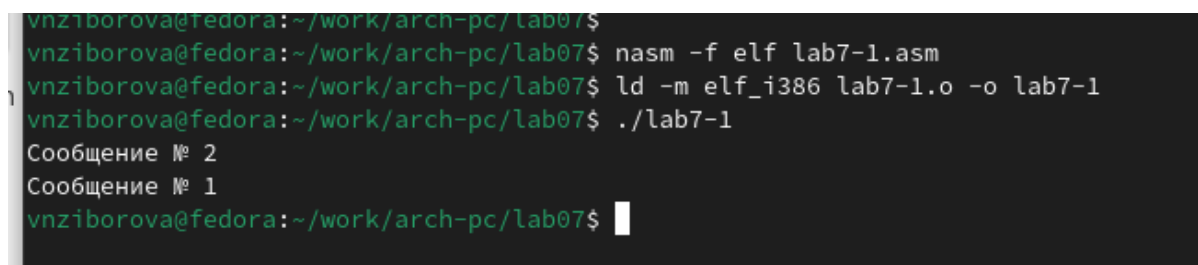
_label1:
mov eax, msg1
call sprintLF
jmp _end

_label2:
mov eax, msg2
call sprintLF
jmp _label1

_label3:
mov eax, msg3
call sprintLF

_end:
call quit
```

Рис. 2.3: Измененная программа в lab7-1.asm



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

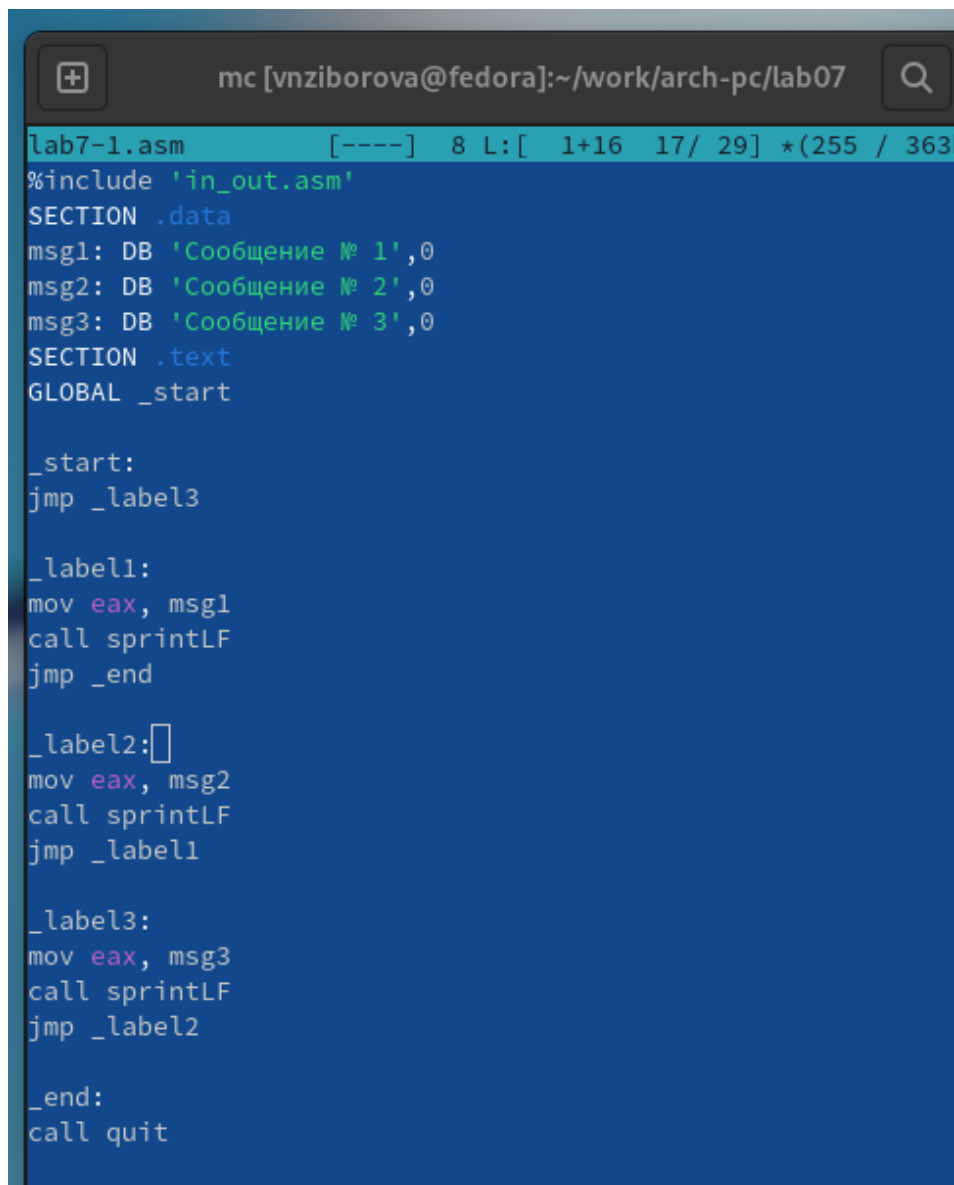
Рис. 2.4: Запуск измененной программы

Затем изменила текст программы так, чтобы она выводила:

Сообщение № 3

Сообщение № 2

Сообщение № 1



```
lab7-1.asm      [----]  8 L:[ 1+16 17/ 29] *(255 / 363)
#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
```

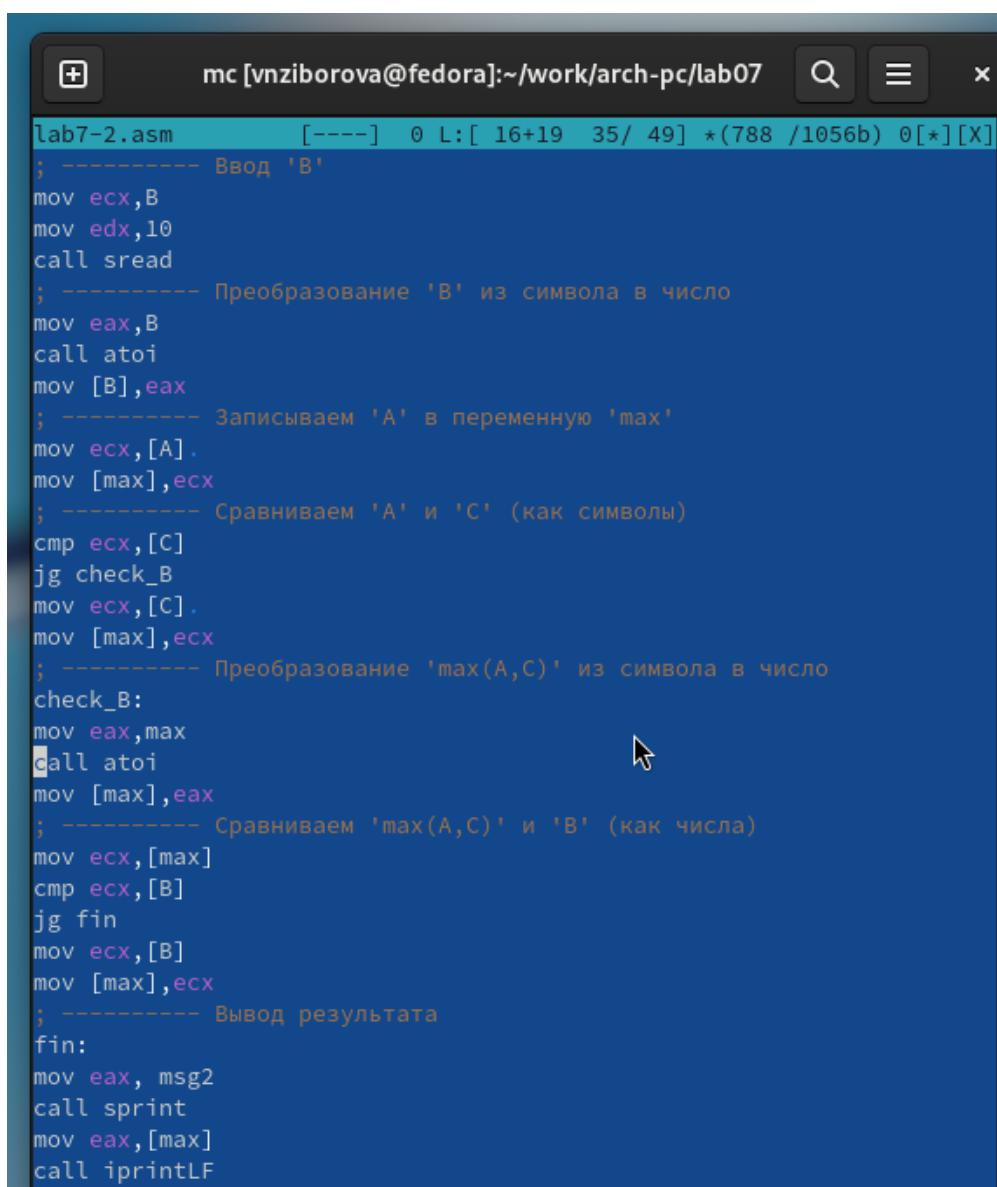
Рис. 2.5: Финальная версия программы lab7-1.asm

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

Рис. 2.6: Запуск программы с новой последовательностью

Инструкция `jmp` всегда выполняет переход, но в программах часто нужно делать условные переходы, когда действие зависит от выполнения условия. Рассмотрим пример с определением наибольшего числа из трех: А, В и С. Значения А и С заданы в программе, В вводится с клавиатуры.

Скомпилировала исполняемый файл и протестировала для разных значений В.



```
mc [vnziborova@fedora]:~/work/arch-pc/lab07
lab7-2.asm [----] 0 L: [ 16+19 35/ 49] *(788 /1056b) 0[*] [X]
; ----- 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 для нахождения максимума в lab7-2.asm

```
vnziborova@fedora:~/work/arch-pc/lab07$  
vnziborova@fedora:~/work/arch-pc/lab07$  
vnziborova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm  
vnziborova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2  
vnziborova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 5  
Наибольшее число: 50  
vnziborova@fedora:~/work/arch-pc/lab07$ ./lab7-2  
Введите В: 55  
Наибольшее число: 55  
vnziborova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.8: Запуск программы для нахождения максимума

2.2 Изучение структуры файла листинга

NASM обычно создает только объектный файл, но с ключом `-l` можно получить файл листинга.

Создала файл листинга для программы из `lab7-2.asm`.

```
lab7-2.lst [----] 0 L:[192+ 9 201/225] *(12300/13771b) 0032 0x020
17 000000F2 B9[0A000000] mov ecx,B
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 число
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 E8ACFFFFFF call sprintf
47 00000163 A1[00000000] mov eax,[max]
48 00000168 E819FFFFFF call iprintLF
49 0000016D E869FFFFFF call quit
```

Рис. 2.9: Файл листинга lab7-2

Изучила содержимое файла листинга. Приведу пример трех строк:

Строка 203

- 28 - номер строки
- 0000011C - адрес
- 3B0D[39000000] - машинный код
- `cmp ecx,[C]` - команда сравнения регистров ecx и переменной C

Строка 204

- 29 - номер строки
- 00000122 - адрес
- 7F0C - машинный код
- `jg check_B` - условный переход к метке check_B, если >.

Строка 205

- 30 - номер строки
- 00000124 - адрес
- 8B0D[39000000] - машинный код
- **mov ecx,[C]** - копирует значение C в ecx

Открыла lab7-2.asm, удалила операнд в инструкции с двумя операндами, выполнила трансляцию и получила файл листинга с ошибкой.

```
vnziborova@fedora:~/work/arch-pc/lab07$  
vnziborova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
vnziborova@fedora:~/work/arch-pc/lab07$  
vnziborova@fedora:~/work/arch-pc/lab07$ nasm -f elf lab7-2.asm -l lab7-2.lst  
lab7-2.asm:34: error: invalid combination of opcode and operands  
vnziborova@fedora:~/work/arch-pc/lab07$  
vnziborova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.10: Ошибка трансляции lab7-2

```
mc [vnziborova@fedora]:~/work/arch-pc/lab07
lab7-2.lst [----] 0 L:[193+33 226/226] *(13859/13859b) <EOF>
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 число
33 check_B:
34 mov eax,
34 ***** error: invalid combination of opcode and operands
35 00000130 E867FFFFFF call atoi
36 00000135 A3[00000000] mov [max],eax
37 ; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
38 0000013A 8B0D[00000000] mov ecx,[max]
39 00000140 3B0D[0A000000] cmp ecx,[B]
40 00000146 7F0C jg fin
41 00000148 8B0D[0A000000] mov ecx,[B]
42 0000014E 890D[00000000] mov [max],ecx
43 ; ----- Вывод результата
44 fin:
45 00000154 B8[13000000] mov eax,msg2
46 00000159 E8B1FEFFFF call sprint
47 0000015E A1[00000000] mov eax,[max]
48 00000163 E81EFFFFFF call iprintLF
49 00000168 E86EFFFFFF call quit
```

Рис. 2.11: Листинг с ошибкой lab7-2

Объектный файл не создан, но в листинге видно место ошибки.

2.3 Задание для самостоятельной работы

Задание 1: Написать программу нахождения наименьшей из трех целых чисел а, b и с. Значения выбрать из таблицы 7.5

по варианту 6: 79,83,41. Скомпилировать и проверить программу.


```
mc [vnziborova@fedora]:~/work/arch-pc/la
task7-1.asm [----] 0 L: [ 37+ 4 41/ 71] *(605 /1062b) 0
mov eax,msgC
call sprint
mov ecx,C
mov edx,80
call sread.
mov eax,C
call atoi
mov [C],eax...
;-----_algorithm-----
....
mov ecx,[A] ;ecx = A
mov [min],ecx ;min = A.

cmp ecx, [B] ; A&B
j< check_C ; if a<b: goto check_C.
mov ecx, [B]
mov [min], ecx ;else min = B

check_C:
cmp ecx, [C]
j< finish
mov ecx,[C]
mov [min],ecx.

finish:
mov eax,answer
call sprint

mov eax, [min]
call iprintLF

call quit
```

Рис. 2.12: Программа в task7-1.asm

```
vnziborova@fedora:~/work/arch-pc/lab07$
vnziborova@fedora:~/work/arch-pc/lab07$ nasm -f elf task7-1.asm
vnziborova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task7-1.o -o task7-1
vnziborova@fedora:~/work/arch-pc/lab07$ ./task7-1
Input A: 79
Input B: 83
Input C: 41
Smallest: 41
vnziborova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.13: Запуск task7-1.asm

Задание 2: Написать программу, которая вычисляет значение функции $f(x)$ для введенных с клавиатуры значений x и a . Формулу $f(x)$ выбрать из таблицы 7.6 для варианта 6:

$$\begin{cases} x + a, x = a \\ 5x, x \neq a \end{cases}$$

Скомпилировать и протестировать для значений из таблицы.

```

mc [vnziborova@fedora]:~/work/arch-pc/lab07
task7-2.asm  [----] 13 L: [ 18+29 47/ 52] *(693 / 740b) 0010 0x00A
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...

;-----algorithm-----

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

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

second:
mov eax,[X]
mov ebx,5
mul ebx
call iprintLF.
call quit

```

Рис. 2.14: Программа для $f(x)$ в task7-2.asm

```
vnziborova@fedora:~/work/arch-pc/lab07$  
vnziborova@fedora:~/work/arch-pc/lab07$ nasm -f elf task7-2.asm  
vnziborova@fedora:~/work/arch-pc/lab07$ ld -m elf_i386 task7-2.o -o task7-2  
vnziborova@fedora:~/work/arch-pc/lab07$ ./task7-2  
Input A: 2  
Input X: 2  
4  
vnziborova@fedora:~/work/arch-pc/lab07$ ./task7-2  
Input A: 1  
Input X: 2  
10  
vnziborova@fedora:~/work/arch-pc/lab07$
```

Рис. 2.15: Запуск task7-2.asm

3 Выводы

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