

Лабораторная работа №7

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

Борисенкова София Павловна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение задания для самостоятельной работы	16
4	Выводы	20

Список иллюстраций

2.1	Создание рабочей директории и файла lab7-1.asm	6
2.2	Вставка кода из файла листинга 7.1	7
2.3	Копирование файла in_out.asm в рабочую директорию	7
2.4	Сборка программы из файла lab7-1.asm и её запуск	8
2.5	Изменение файла lab7-1.asm согласно листингу 7.2	8
2.6	Повторная сборка программы из файла lab7-1.asm и её запуск . . .	9
2.7	Редактирование файла lab7-1.asm	9
2.8	Повторная сборка программы из файла lab7-1.asm и её запуск . . .	10
2.9	Создание второго файла: lab7-2.asm	10
2.10	Запись кода из листинга 7.3 в файл lab7-2.asm	11
2.11	сборка программы из файла lab7-2.asm и её запуск	12
2.12	Открытие файла листинга в текстовом редакторе	12
2.13	Вид файла листинга	13
2.14	Изменение исходного файла	14
2.15	Вывод ошибки при сборке объектного файла	15
2.16	Отображение ошибки в листинге	15
3.1	Код первого файла самостоятельной работы	17
3.2	Сборка и запуск программы первого задания	17
3.3	Код второго файла самостоятельной работы	18
3.4	Сборка и тестирование второго файла самостоятельной работы . .	19

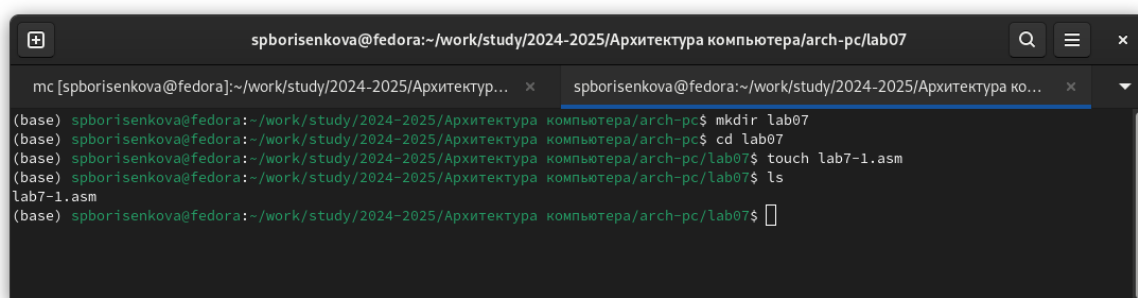
Список таблиц

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

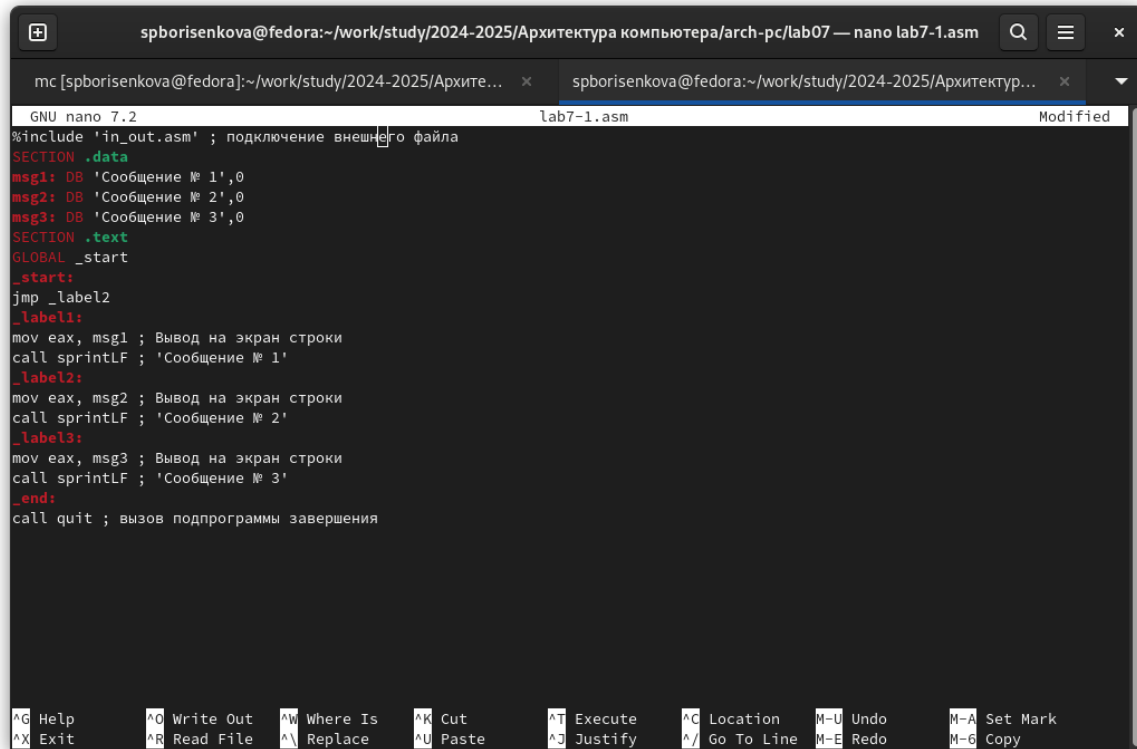
Для начала выполнения лабораторной работы необходимо создать рабочую папку lab07 и файл lab7-1.asm (рис. 2.1):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектур... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура ко... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ mkdir lab07
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cd lab07
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ls
lab7-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.1: Создание рабочей директории и файла lab7-1.asm

Вставим код в файл lab7-1.asm из файла листинга (рис. 2.2):

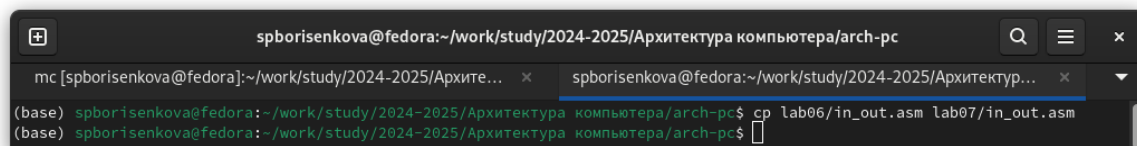


```
GNU nano 7.2 lab7-1.asm Modified
mc [spborisenkova@fedora:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектур... x
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg1: DB 'Сообщение № 1',0
msg2: DB 'Сообщение № 2',0
msg3: DB 'Сообщение № 3',0
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

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo     M-A Set Mark
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_/ Go To Line M-E Redo     M-G Copy
```

Рис. 2.2: Вставка кода из файла листинга 7.1

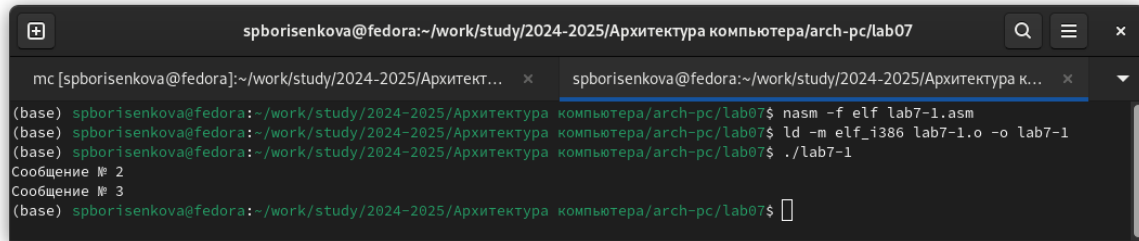
Теперь скопируем файл `in_out.asm` из рабочей директории прошлой лабораторной работы (рис. 2.3):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$ cp lab06/in_out.asm lab07/in_out.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc$
```

Рис. 2.3: Копирование файла `in_out.asm` в рабочую директорию

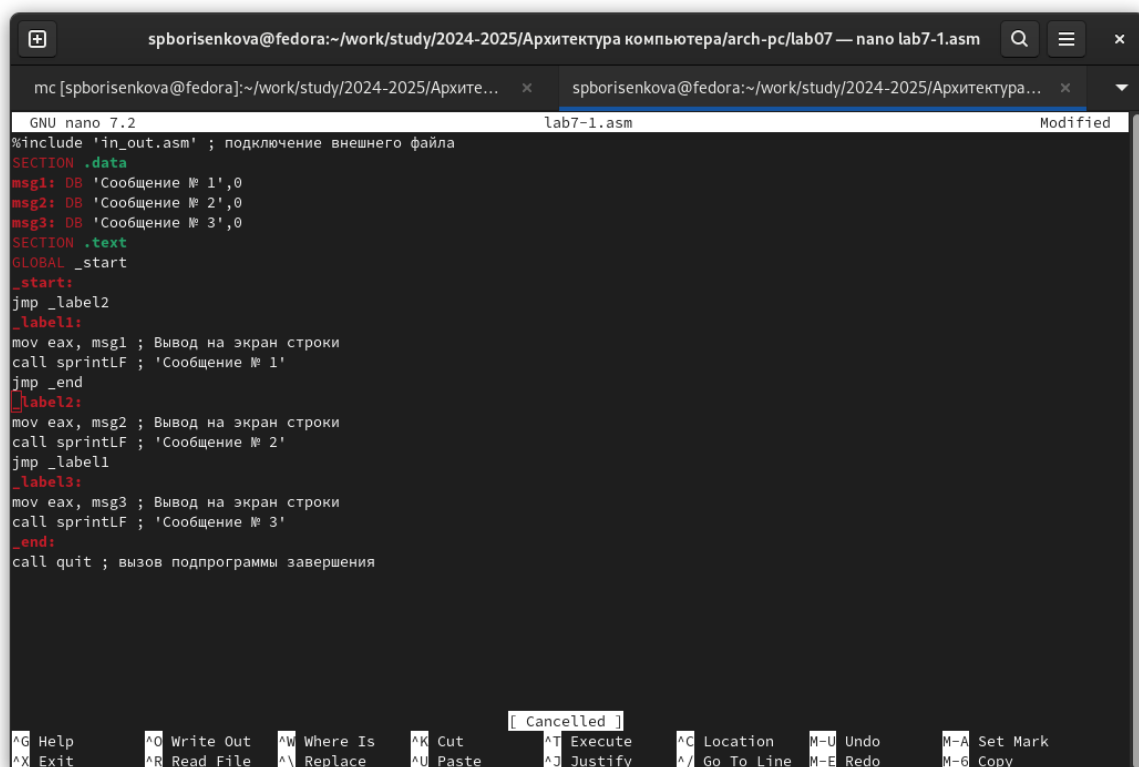
Теперь соберём программу из файла `lab7-1.asm` и запустим её (рис. 2.4):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитек... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура к... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 3
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.4: Сборка программы из файла lab7-1.asm и её запуск

Изменим файл lab7-1.asm согласно листингу 7.2 (рис. 2.5):



```
GNU nano 7.2 lab7-1.asm Modified
%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

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute  ^C Location  M-U Undo     M-A Set Mark
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify  ^_ Go To Line M-E Redo     M-G Copy
```

Рис. 2.5: Изменение файла lab7-1.asm согласно листингу 7.2

Снова соберём программу и запустим её (рис. 2.6):


```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 2
Сообщение № 1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

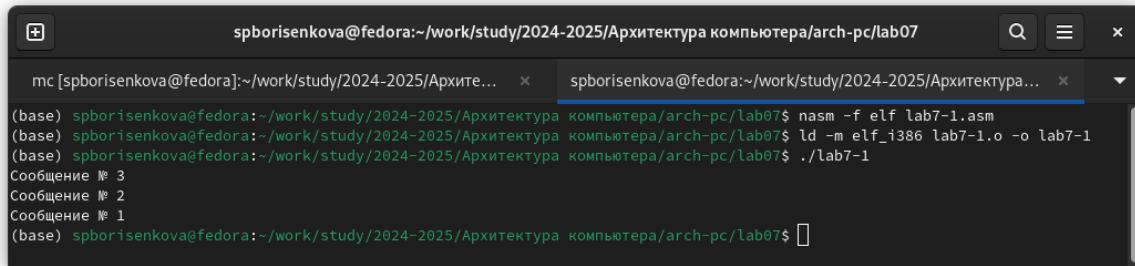
Рис. 2.6: Повторная сборка программы из файла lab7-1.asm и её запуск

Теперь сделаем так, чтобы код выводил сообщения в от третьего к первому. Для этого внесём в код следующие изменения (рис. 2.7):

```
GNU nano 7.2 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
```

Рис. 2.7: Редактирование файла lab7-1.asm

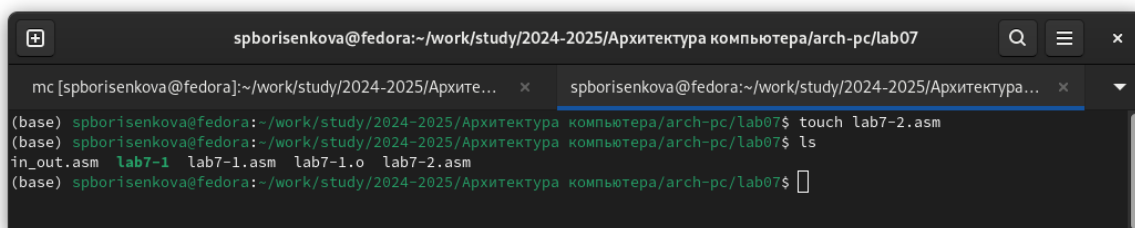
И запустим её, предварительно собрав (рис. 2.8):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-1.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 lab7-1.o -o lab7-1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-1
Сообщение № 3
Сообщение № 2
Сообщение № 1
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.8: Повторная сборка программы из файла lab7-1.asm и её запуск

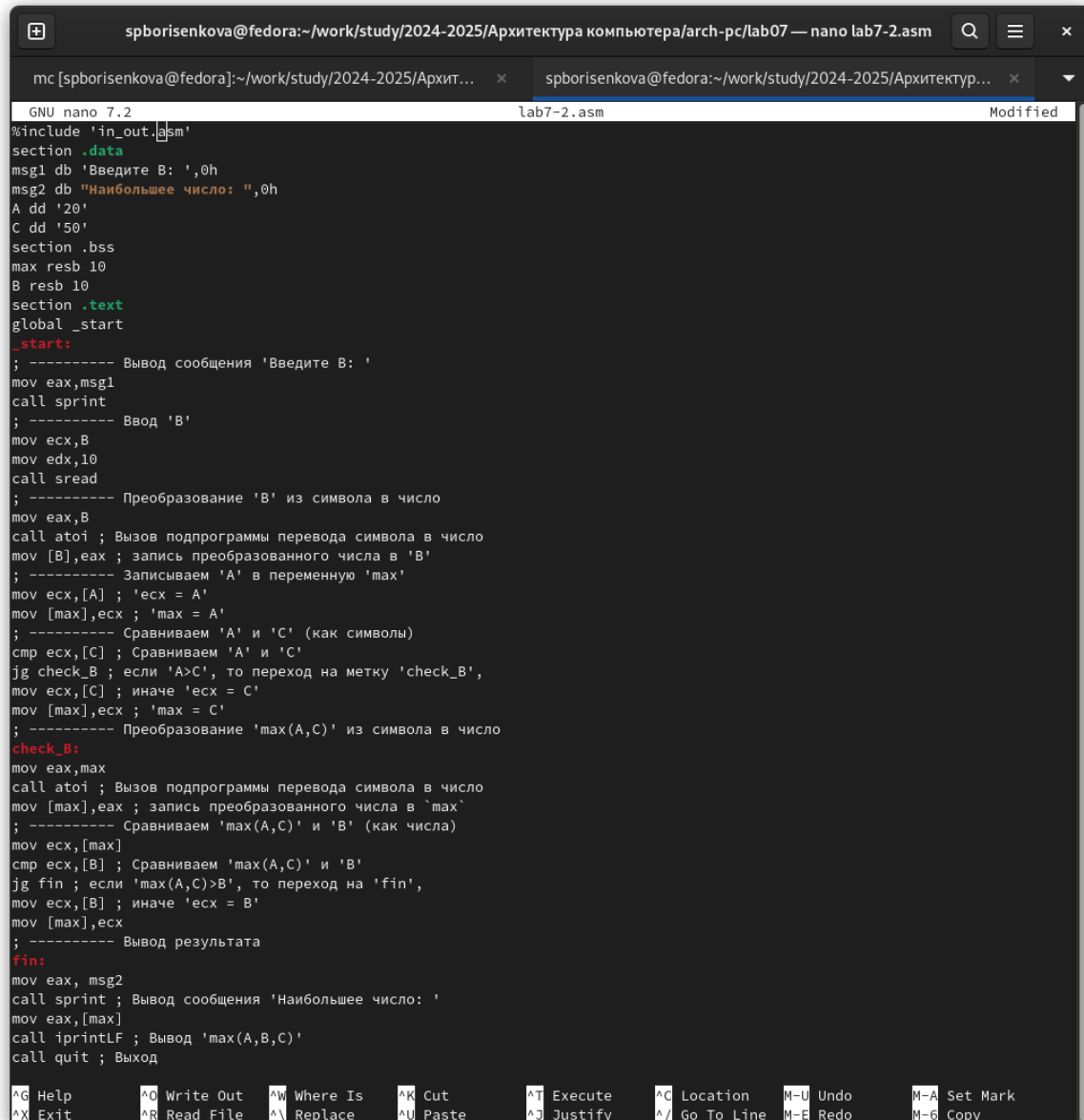
Теперь создадим файл lab7-2.asm (рис. 2.9):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora:~/work/study/2024-2025/Архите... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ touch lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ls
in_out.asm lab7-1 lab7-1.asm lab7-1.o lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.9: Создание второго файла: lab7-2.asm

Запишем код из листинга 7.3 в файл lab7-2.asm (рис. 2.10):



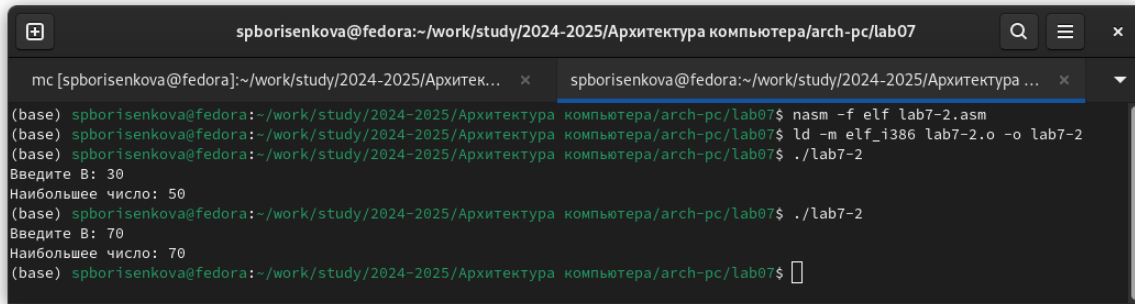
The screenshot shows a terminal window with the nano text editor open. The title bar indicates the user is 'spborisenkova' on a 'fedora' system, working in the directory '~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07'. The editor is editing a file named 'lab7-2.asm'. The code is written in assembly language and includes comments in Russian. It defines data for messages and constants, reserves space for variables B and max, and implements a logic to find the maximum of three input numbers A, B, and C. The code uses standard x86 assembly instructions like 'mov', 'call', 'cmp', and 'jg', along with system calls for printing and reading. The bottom of the window shows a keyboard shortcuts menu.

```
GNU nano 7.2 lab7-2.asm Modified
%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,10
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
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax, msg2
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^_ Replace    ^U Paste      ^J Justify    ^_ Go To Line M-E Redo      M-G Copy
```

Рис. 2.10: Запись кода из листинга 7.3 в файл lab7-2.asm

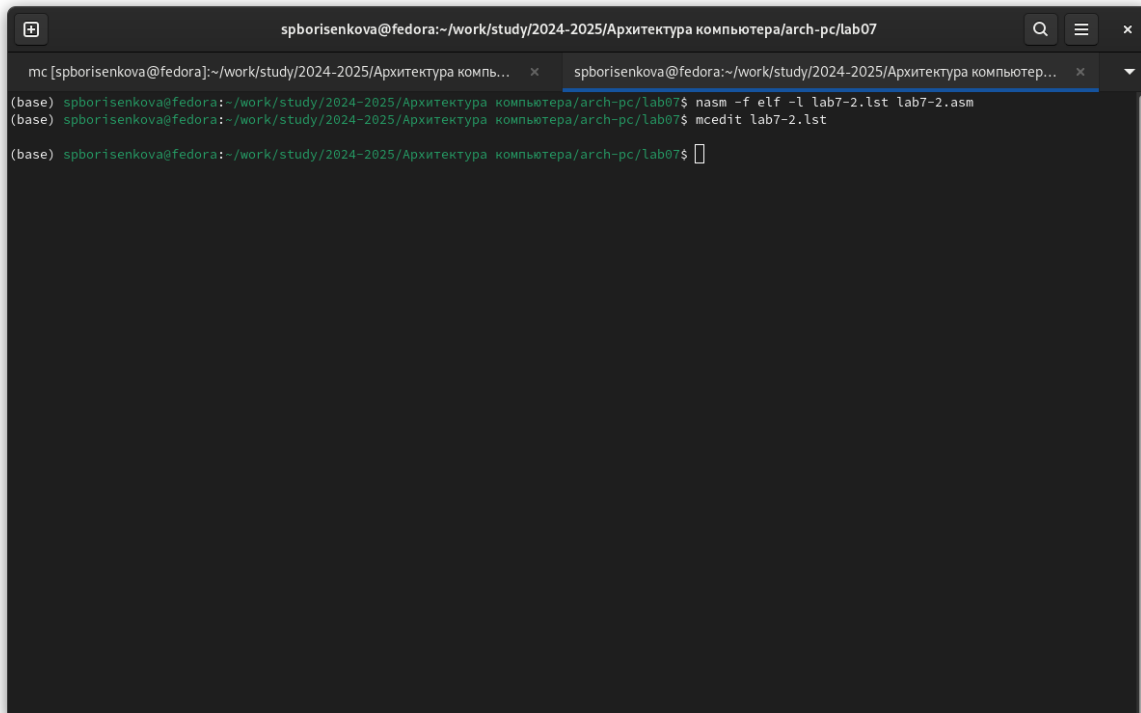
И запустим его, предварительно собрав (рис. 2.11):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитек... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура ... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 lab7-2.o -o lab7-2
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 30
Наибольшее число: 50
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./lab7-2
Введите B: 70
Наибольшее число: 70
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.11: сборка программы из файла lab7-2.asm и её запуск

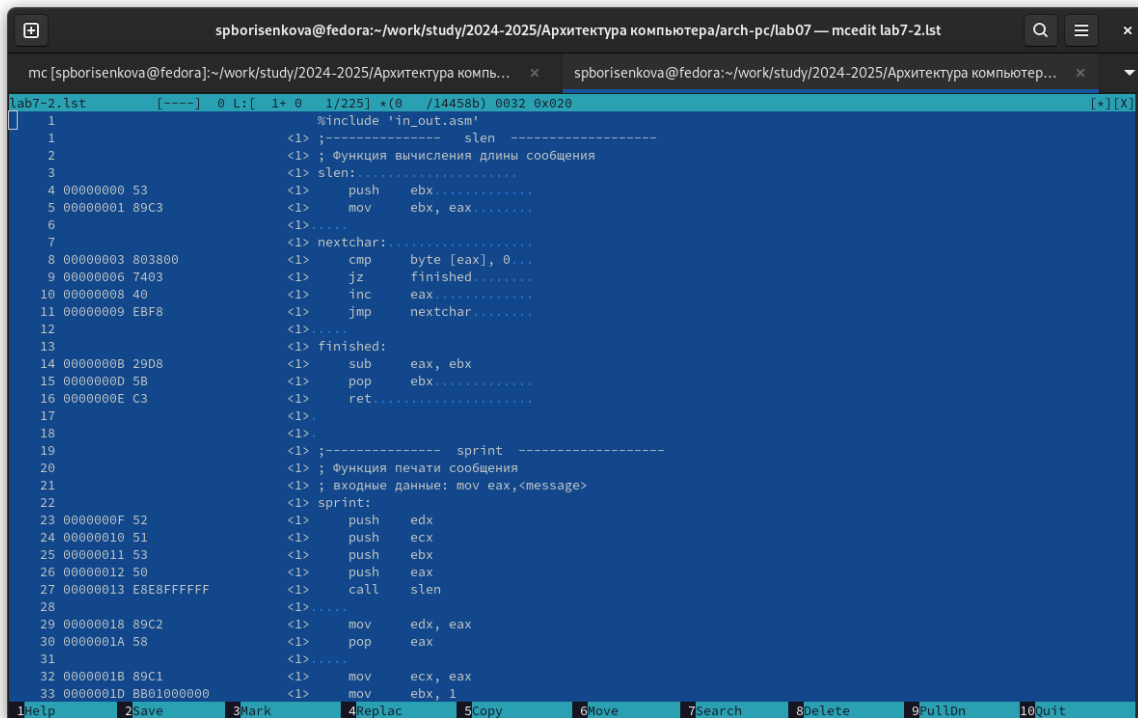
Теперь попробуем создать файл листинга при сборке файла lab7-2.asm и посмотрим, как выглядит файл листинга изнутри. Для этого откроем его в mcedit (рис. 2.12):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектура компь... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьют... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mcedit lab7-2.lst
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.12: Открытие файла листинга в текстовом редакторе

Открыв его, мы видим следующее (рис. 2.13):



```
lab7-2.lst [----] 0 L: [ 1+ 0 1/225] *(0 /14458b) 0032 0x020 [*] [X]
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 00000003 803800 <1> nextchar:
9 00000006 7403 <1> cmp byte [eax], 0
10 00000008 40 <1> jz finished
11 00000009 EBF8 <1> inc eax
12 <1> jmp nextchar
13 <1> finished:
14 0000000B 29D8 <1> sub eax, ebx
15 0000000D 5B <1> pop ebx
16 0000000E C3 <1> ret
17 <1>
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 BB01000000 <1> mov ebx, 1
```

Рис. 2.13: Вид файла листинга

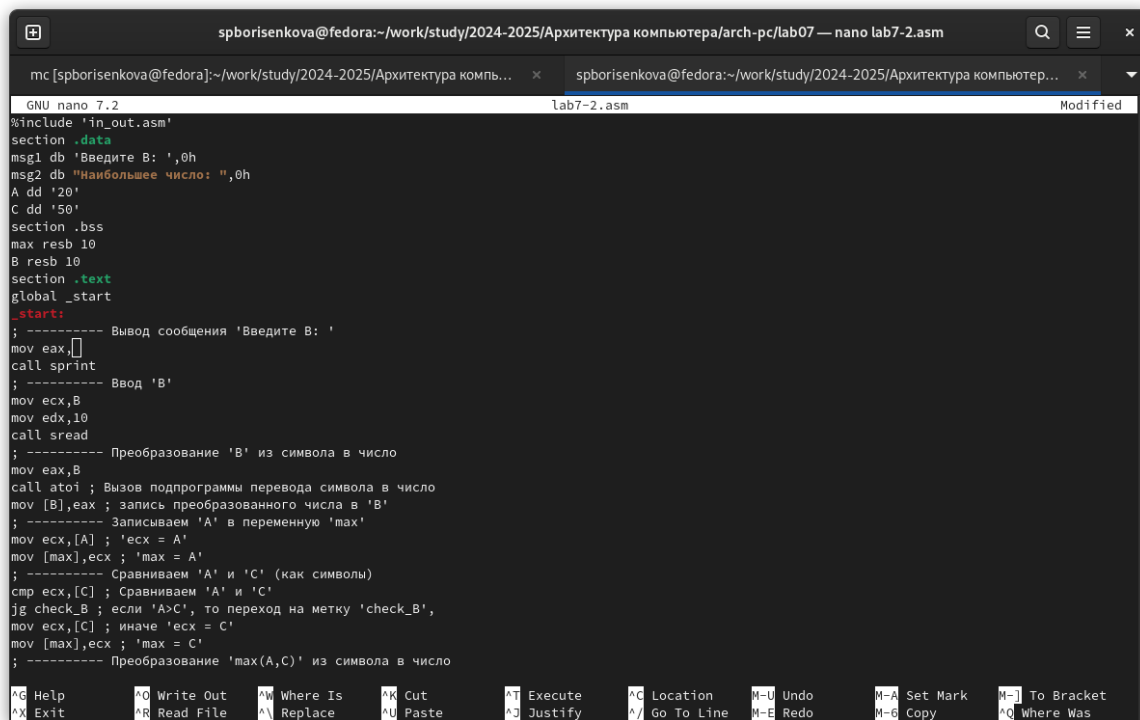
Разберём несколько строк файла листинга:

1. Строка под номером 21 перемещает содержимое В в регистр еах. Адрес указывается сразу после номера. Следом идёт машинный код, который представляет собой исходную ассемблированную строку в виде шестнадцатиричной системы. Далее идёт исходный код

2. 22-ая строка отвечает за вызов функции atoi. Она также имеет адрес и машинный код

3. Строка 23 отвечает за запись значения в регистре еах в В.

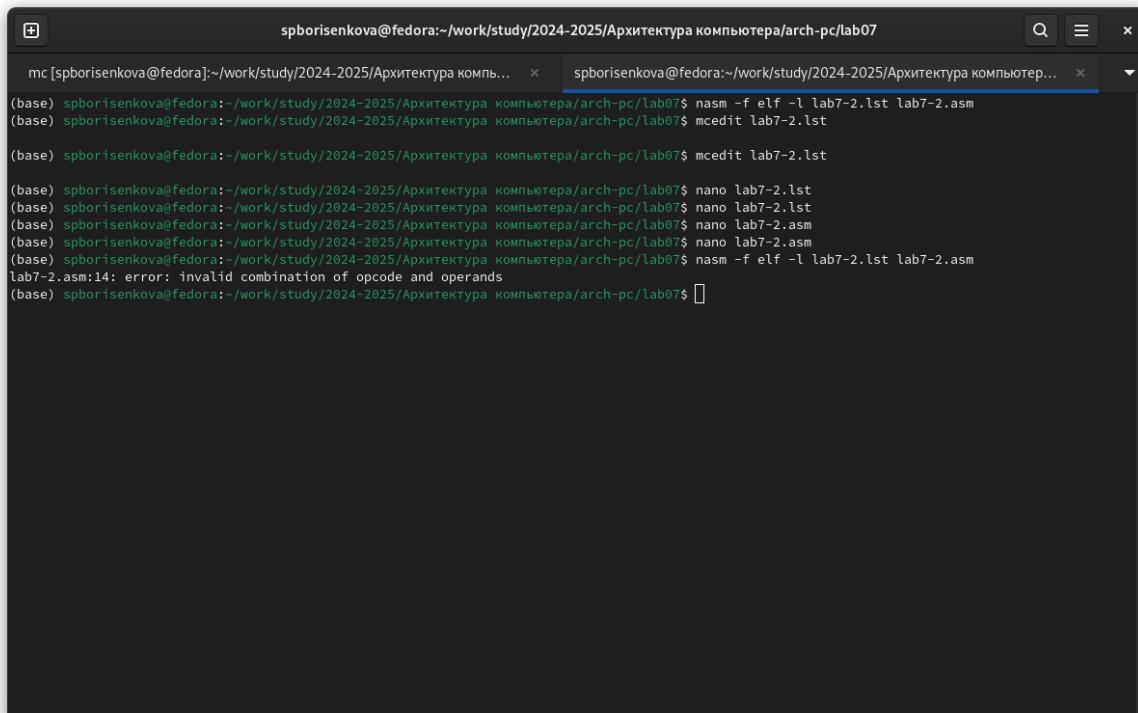
Теперь попробуем намеренно допустить ошибку в нашем коде, убрав у команды один из операндов (рис. 2.14):



```
GNU nano 7.2 lab7-2.asm Modified
#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,0
call sprint
; ----- Ввод 'B'
mov ecx,B
mov edx,10
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
```

Рис. 2.14: Изменение исходного файла

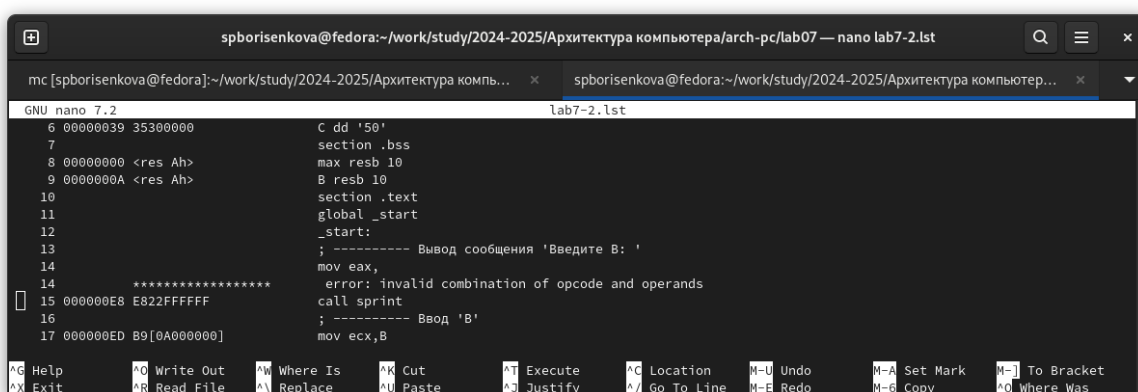
И попробуем собрать файл с ошибкой, генерируя файл листинга (рис. 2.15):



```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектура компь... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютер... x
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mcedit lab7-2.lst
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ mcedit lab7-2.lst
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nano lab7-2.lst
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nano lab7-2.lst
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nano lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nano lab7-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf -l lab7-2.lst lab7-2.asm
lab7-2.asm:14: error: invalid combination of opcode and operands
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 2.15: Вывод ошибки при сборке объектного файла

Мы видим, что объектный файл не создался, однако появился файл листинга. Теперь зайдём в файл листинга, и посмотрим, отображается ли в нём ошибка (рис. 2.16):



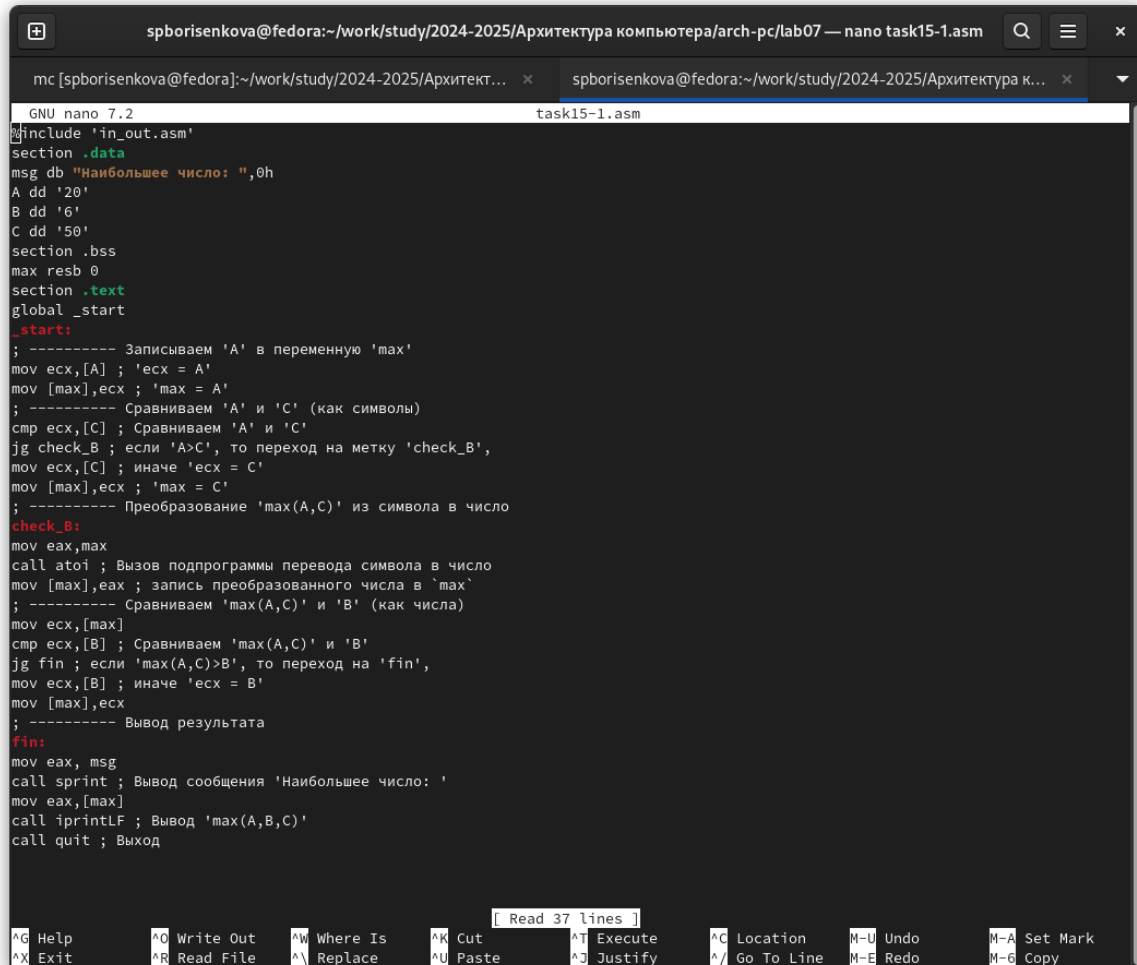
```
spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07 — nano lab7-2.lst
mc [spborisenkova@fedora]:~/work/study/2024-2025/Архитектура компь... x spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютер... x
GNU nano 7.2 lab7-2.lst
6 00000039 35300000 C dd '50'
7 section .bss
8 00000000 <res Ah> max resb 10
9 0000000A <res Ah> B resb 10
10 section .text
11 global _start
12 _start:
13 ; ----- Вывод сообщения 'Введите B: '
14 mov eax,
15 error: invalid combination of opcode and operands
16 call sprint
17 000000E8 E822FFFFFF ; ----- Ввод 'B'
18 mov ecx,B
19
```

Рис. 2.16: Отображение ошибки в листинге

Как видим, в листинге прописана ошибка

3 Выполнение задания для самостоятельной работы

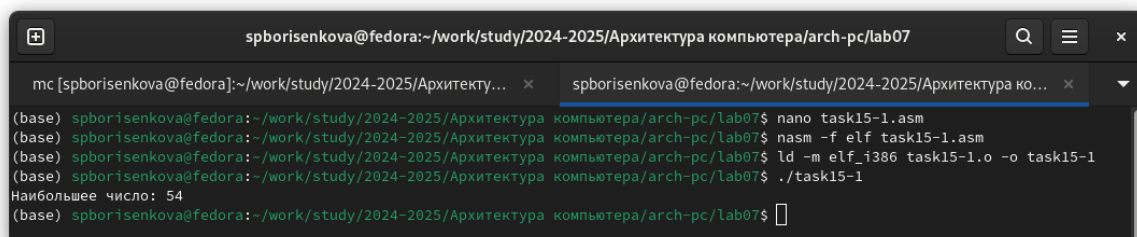
Создадим файл для выполнения самостоятельной работы. Мой вариант - 15.
Напишем код для выполнения задания. Код выглядит так (рис. 3.1):



```
GNU nano 7.2 task15-1.asm
#include 'in_out.asm'
section .data
msg db "Наибольшее число: ",0h
A dd '20'
B dd '6'
C dd '50'
section .bss
max resb 0
section .text
global _start
_start:
; ----- 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
mov [max],eax ; запись преобразованного числа в 'max'
; ----- Сравниваем 'max(A,C)' и 'B' (как числа)
mov ecx,[max]
cmp ecx,[B] ; Сравниваем 'max(A,C)' и 'B'
jg fin ; если 'max(A,C)>B', то переход на 'fin',
mov ecx,[B] ; иначе 'ecx = B'
mov [max],ecx
; ----- Вывод результата
fin:
mov eax,msg
call sprint ; Вывод сообщения 'Наибольшее число: '
mov eax,[max]
call iprintLF ; Вывод 'max(A,B,C)'
call quit ; Выход
```

Рис. 3.1: Код первого файла самостоятельной работы

Соберём, запустим его и посмотрим на результат (рис. 3.2):

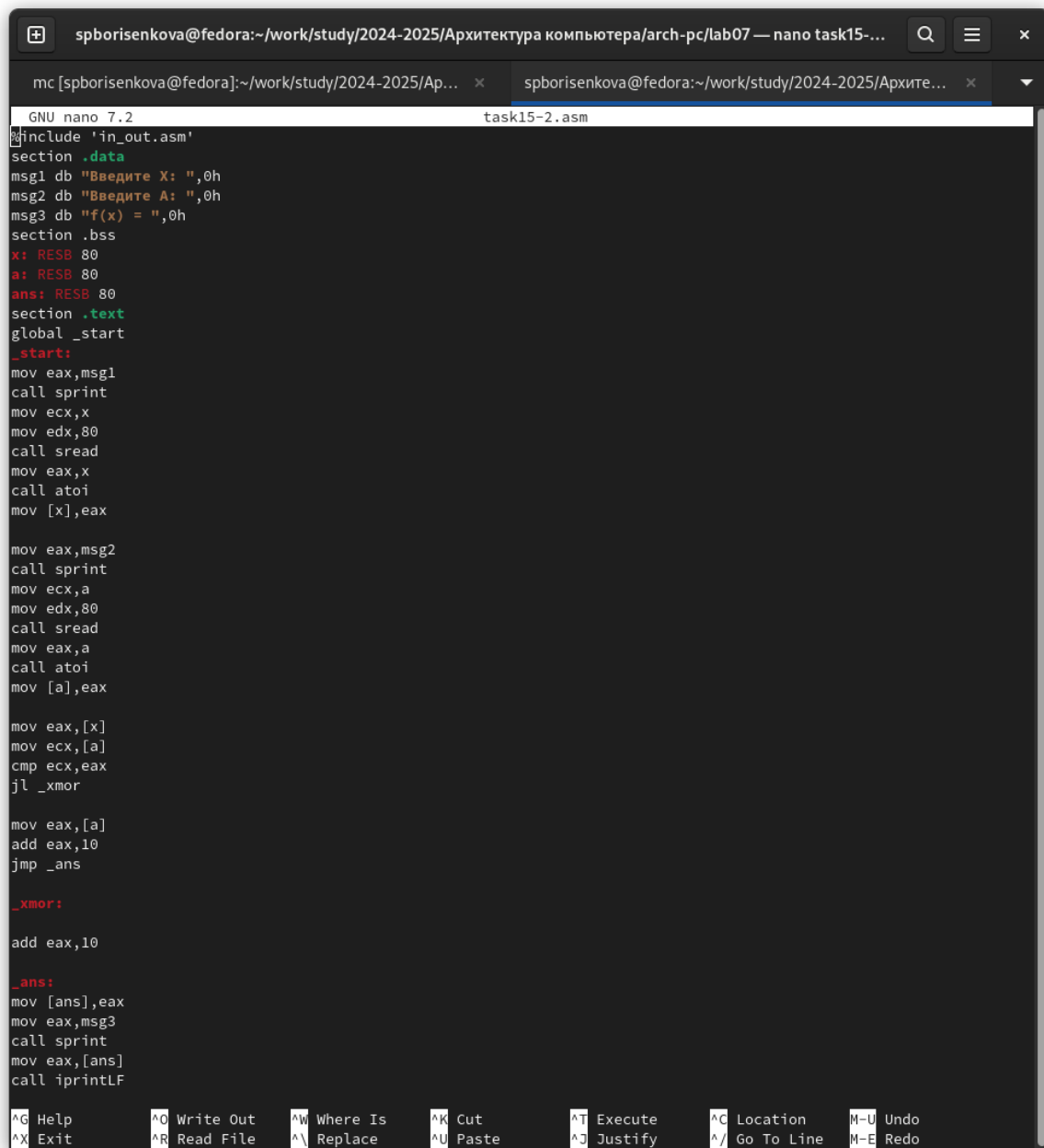


```
(base) spborisenkova@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nano task15-1.asm
(base) spborisenkova@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf task15-1.asm
(base) spborisenkova@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 task15-1.o -o task15-1
(base) spborisenkova@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./task15-1
Наибольшее число: 54
(base) spborisenkova@fedora: ~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 3.2: Сборка и запуск программы первого задания

Теперь создадим второй файл самостоятельной работы для второго задания.

Код будет выглядеть так (рис. 3.3):



```
GNU nano 7.2 task15-2.asm
#include 'in_out.asm'
section .data
msg1 db "Введите X: ",0h
msg2 db "Введите A: ",0h
msg3 db "f(x) = ",0h
section .bss
x: RESB 80
a: RESB 80
ans: RESB 80
section .text
global _start
_start:
mov eax,msg1
call sprint
mov ecx,x
mov edx,80
call sread
mov eax,x
call atoi
mov [x],eax

mov eax,msg2
call sprint
mov ecx,a
mov edx,80
call sread
mov eax,a
call atoi
mov [a],eax

mov eax,[x]
mov ecx,[a]
cmp ecx,eax
jl _xmor

mov eax,[a]
add eax,10
jmp _ans

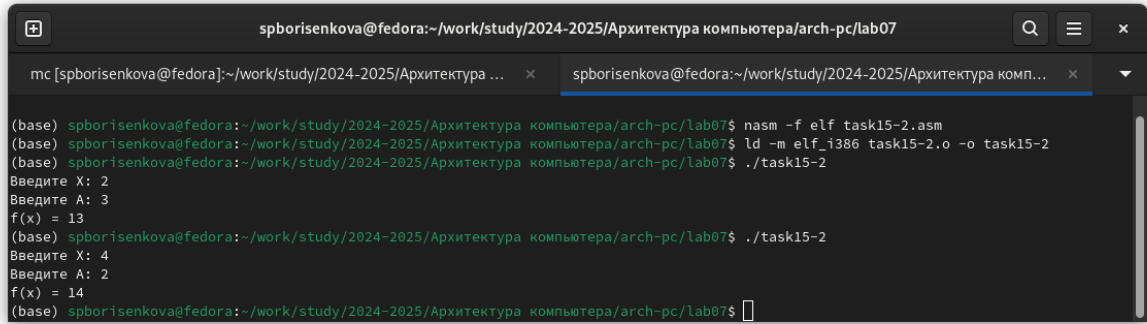
_xmor:
add eax,10

_ans:
mov [ans],eax
mov eax,msg3
call sprint
mov eax,[ans]
call iprintLF
```

Terminal window showing the assembly code for task15-2.asm. The code includes data and bss sections for messages and variables, and a text section for the main logic. It uses standard x86 assembly instructions for string handling and arithmetic. The bottom of the window shows the nano editor's command shortcuts.

Рис. 3.3: Код второго файла самостоятельной работы

Соберём исполняемый файл и запустим его (рис. 3.8):



```
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ nasm -f elf task15-2.asm
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ld -m elf_i386 task15-2.o -o task15-2
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./task15-2
Введите X: 2
Введите A: 3
f(x) = 13
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$ ./task15-2
Введите X: 4
Введите A: 2
f(x) = 14
(base) spborisenkova@fedora:~/work/study/2024-2025/Архитектура компьютера/arch-pc/lab07$
```

Рис. 3.4: Сборка и тестирование второго файла самостоятельной работы

Как видим, программа выполнена успешно

4 Выводы

В результате лабораторной работы были написаны программы, которые используют команды условных и безусловных переходов, были получены навыки работы с этими командами, а также были созданы и успешно прочитаны листинги для некоторых из программ.