

# Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Козин Иван Евгеньевич

## Содержание

### 1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

### 2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

### 3 Теоретическое введение

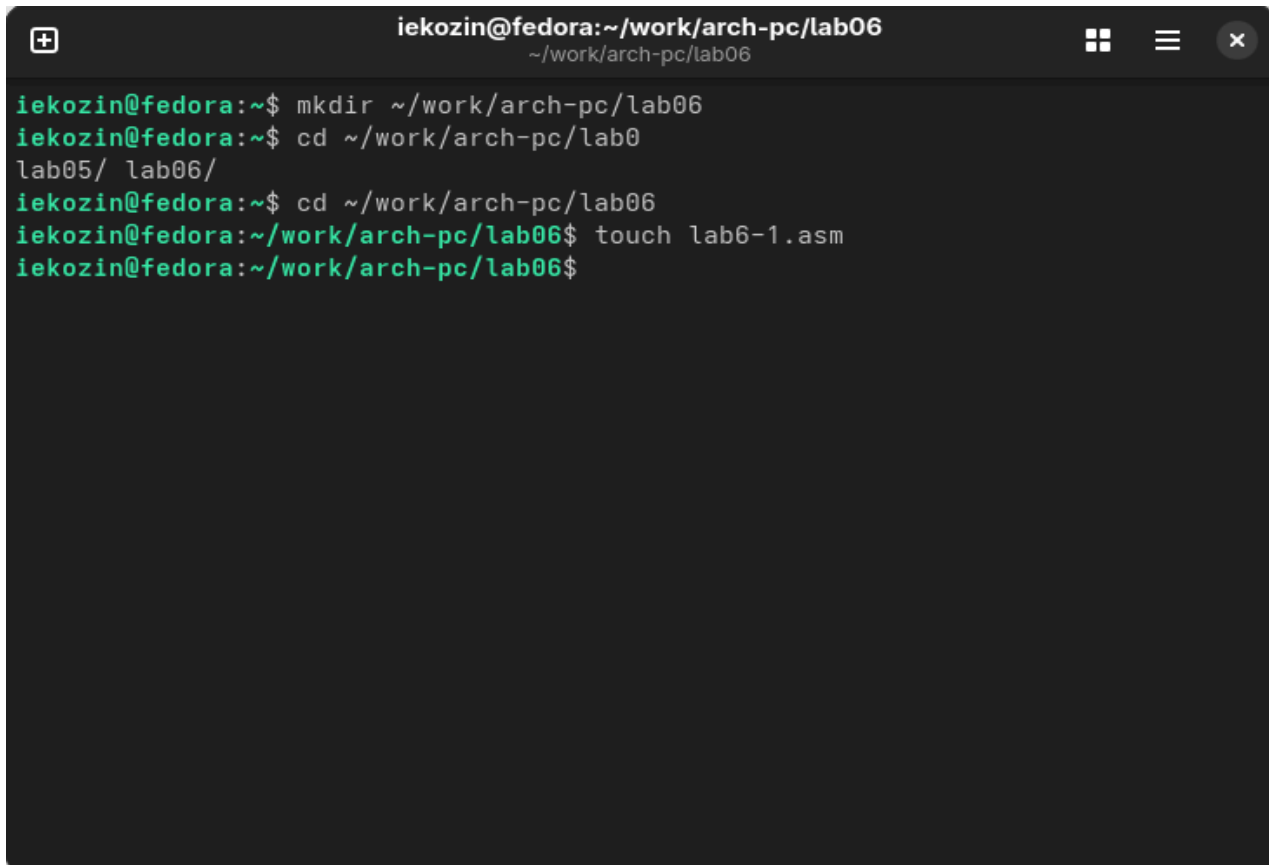
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`. - Непосредственная адресация – значение операнда задается непосредственно в команде, Например: `mov ax,2`. - Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и

выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

## 4 Выполнение лабораторной работы

### 4.1 Символьные и численные данные в NASM

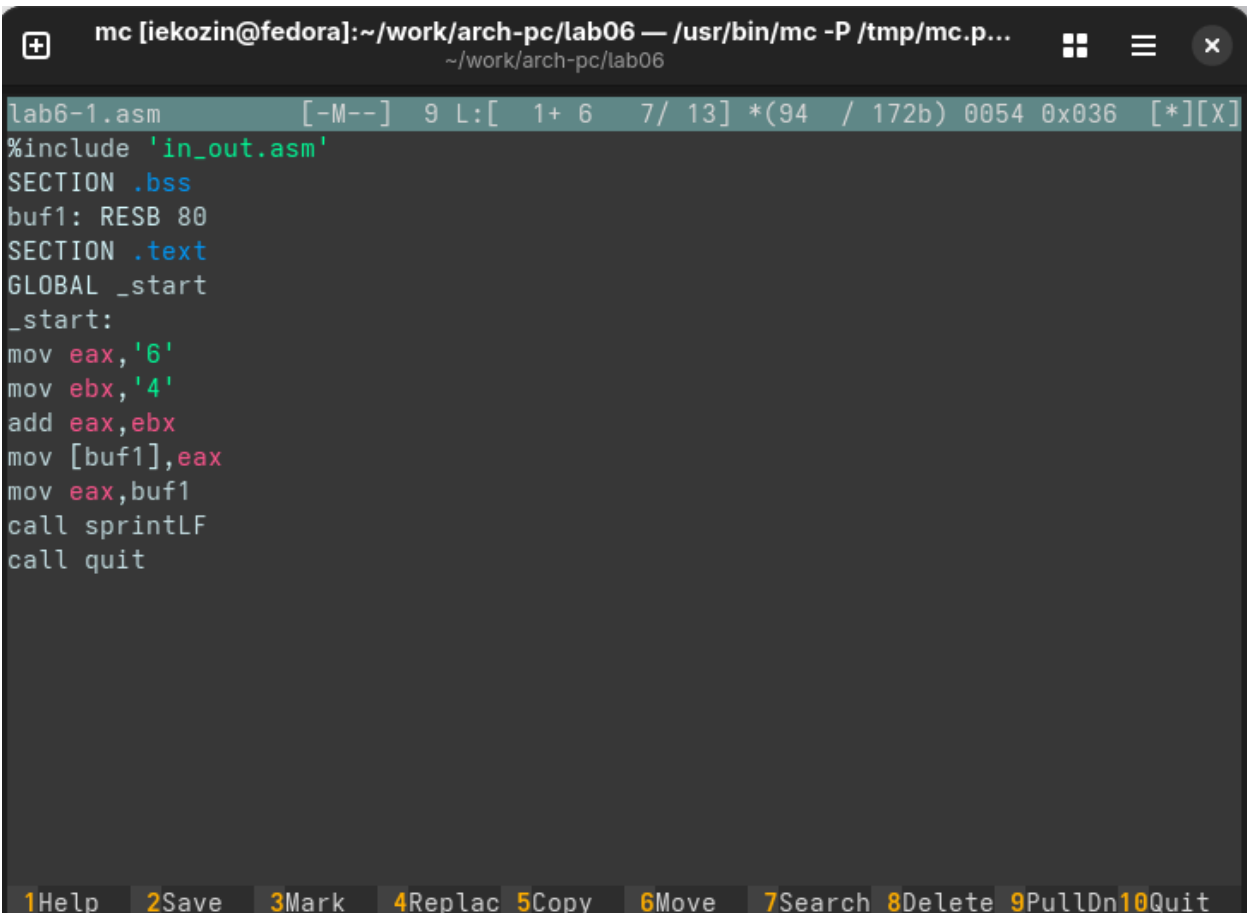
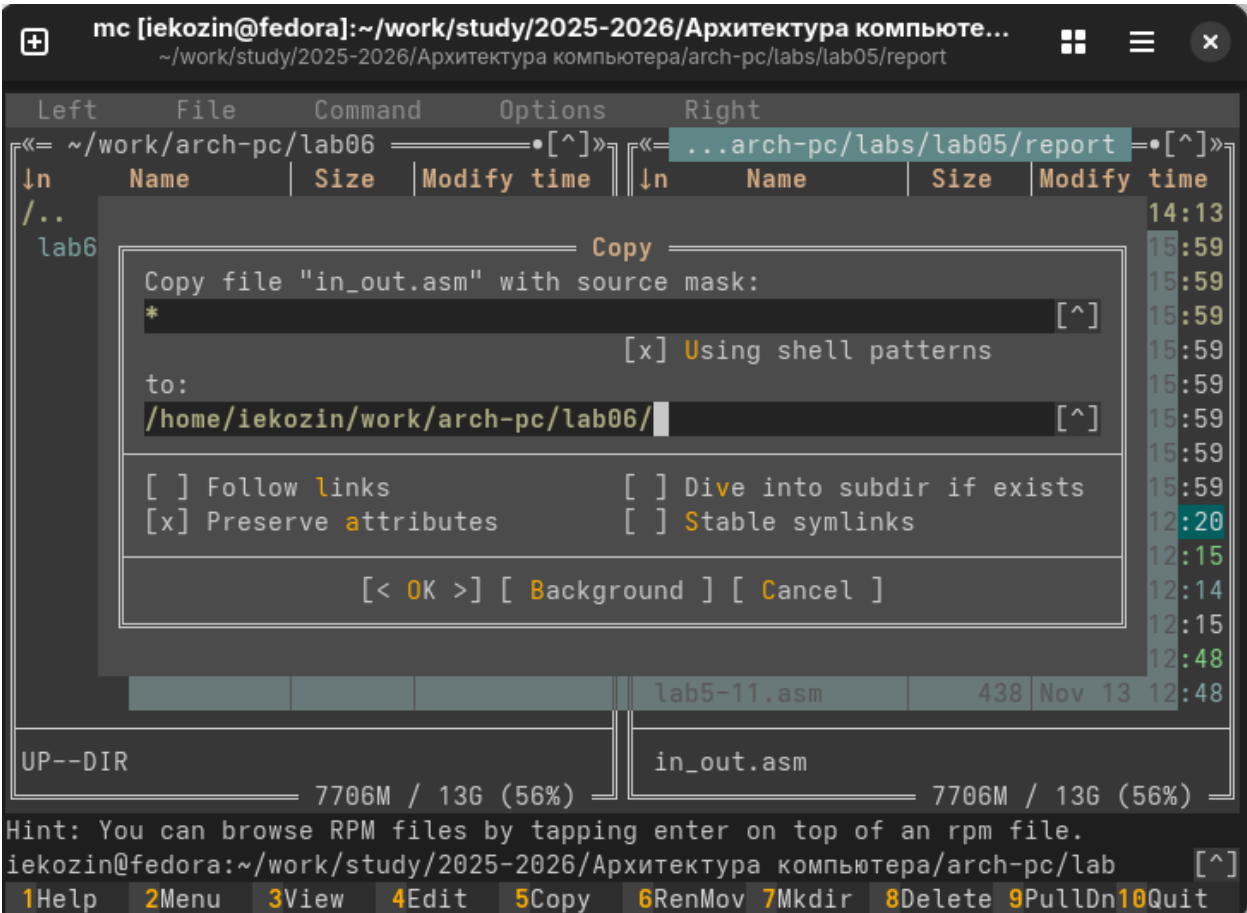
Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл (рис. 1).

A screenshot of a terminal window with a dark background. The window title is 'iekozin@fedora:~/work/arch-pc/lab06'. The terminal shows a series of commands and their outputs: 'mkdir ~/work/arch-pc/lab06', 'cd ~/work/arch-pc/lab0', 'lab05/ lab06/', 'cd ~/work/arch-pc/lab06', and 'touch lab6-1.asm'. The prompt changes from '~\$' to '~/work/arch-pc/lab06\$' after the final command.

```
iekozin@fedora:~$ mkdir ~/work/arch-pc/lab06
iekozin@fedora:~$ cd ~/work/arch-pc/lab0
lab05/ lab06/
iekozin@fedora:~$ cd ~/work/arch-pc/lab06
iekozin@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
iekozin@fedora:~/work/arch-pc/lab06$
```

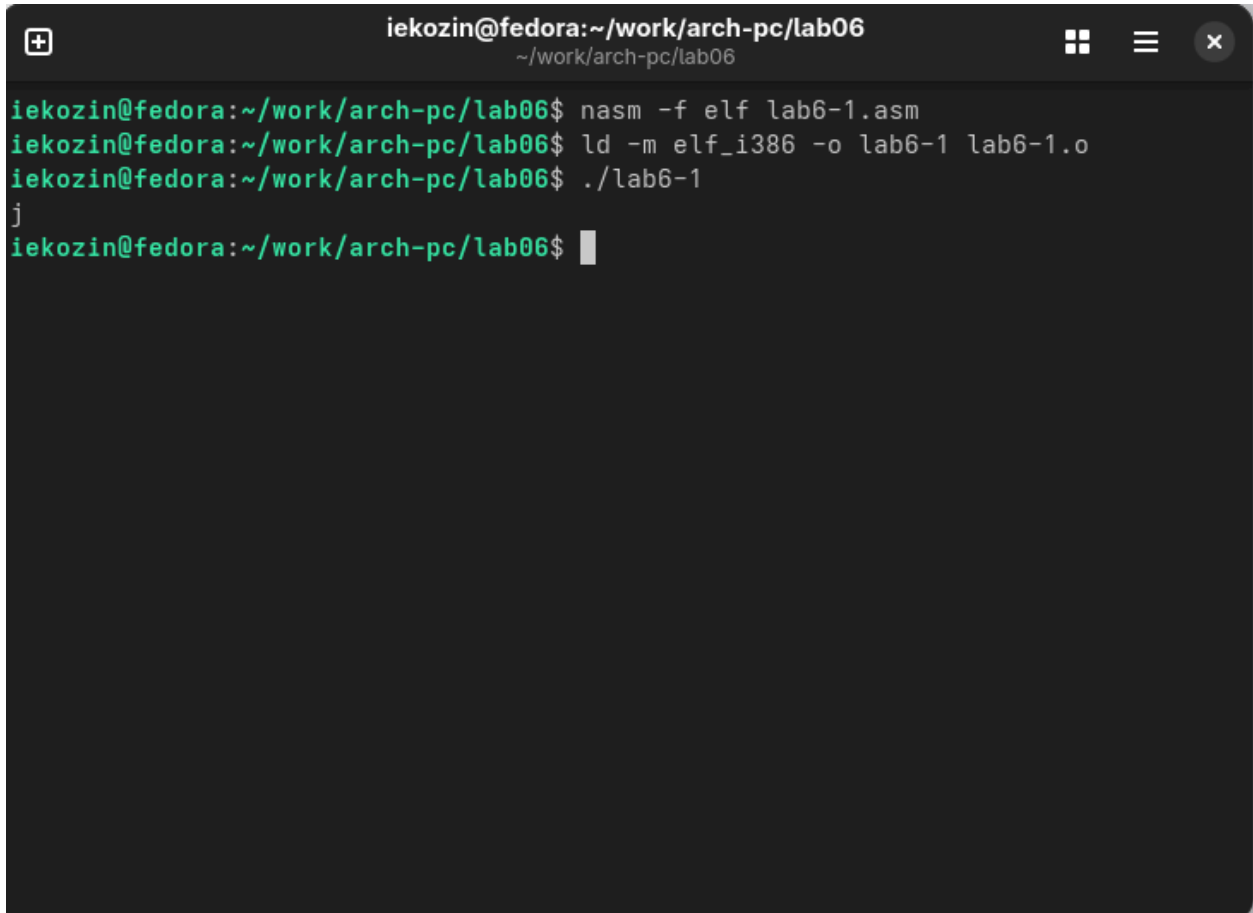
Рис. 1: Создание нового каталога

В созданном файле ввожу программу из листинга (рис. 2), и чтобы программа корректно работала я подключаю `in_out.asm`



*Рис. 2: Сохранение новой программы и подключение файла посредством копирования его из другой директории*

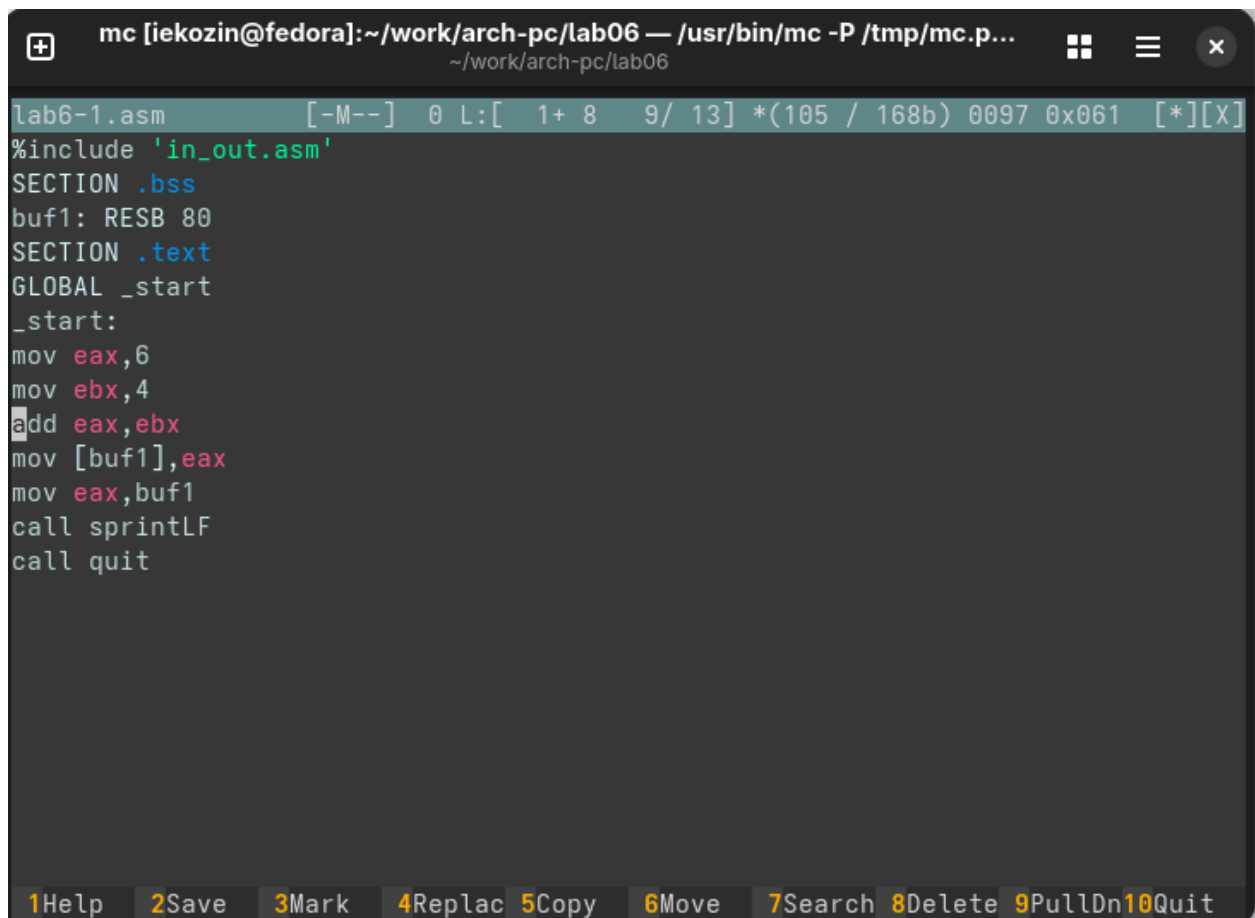
Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII. {#fig:003 width=70%}

A terminal window titled 'iekozin@fedora:~/work/arch-pc/lab06' with a subtitle '~/.work/arch-pc/lab06'. The terminal shows the following commands and output:

```
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
iekozin@fedora:~/work/arch-pc/lab06$
```

*Рис. 3: Запуск изначальной программы*

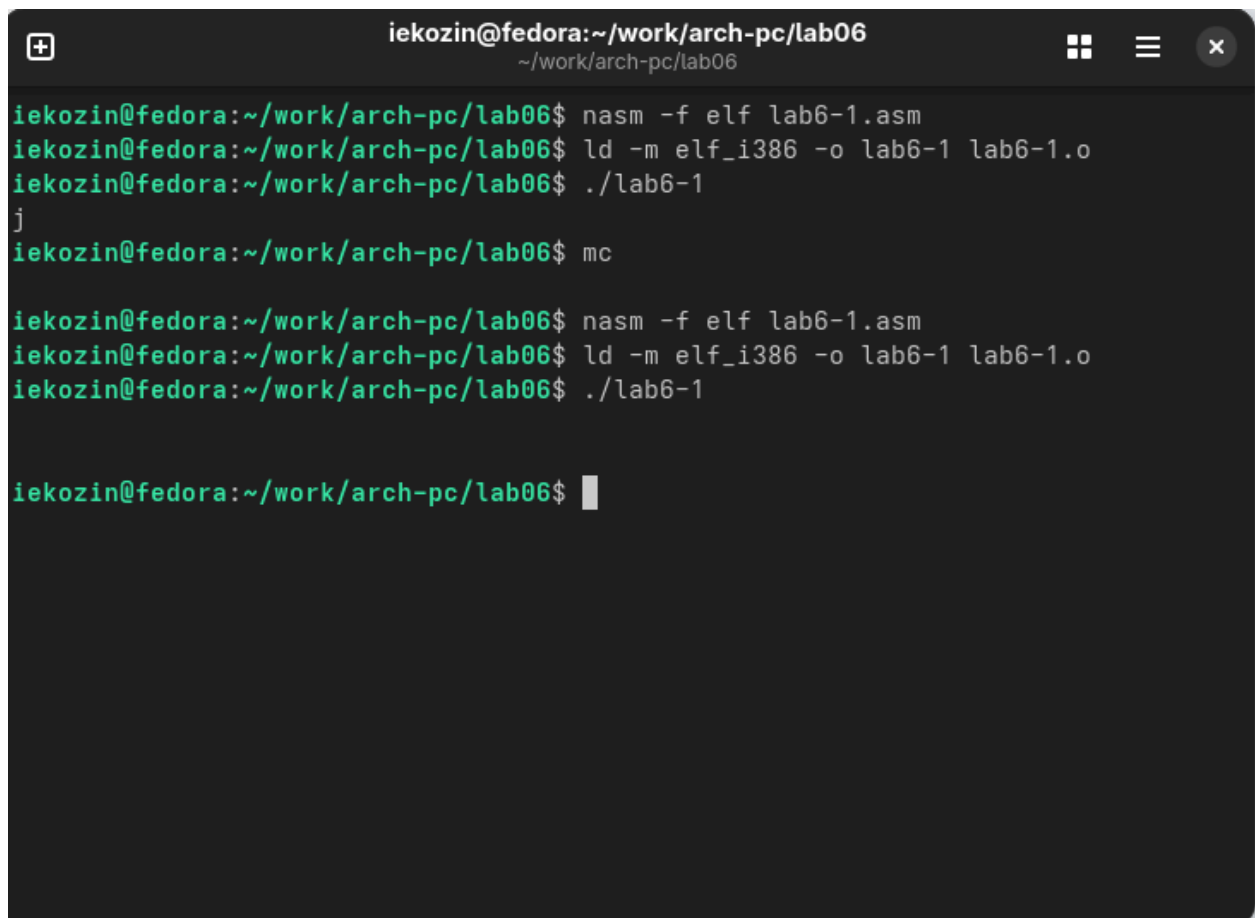
Изменяю текст изначальной программы, убрав кавычки (рис. 4).



```
mc [iekozin@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.p...  
~/work/arch-pc/lab06  
lab6-1.asm [-M--] 0 L: [ 1+ 8 9/ 13] *(105 / 168b) 0097 0x061 [*][X]  
%include 'in_out.asm'  
SECTION .bss  
buf1: RESB 80  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
mov [buf1],eax  
mov eax,buf1  
call sprintf  
call quit  
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 4: Измененная программа

На этот раз программа выдала пустую строку, это связано с тем, что символ 10 означает переход на новую строку (рис. 5).

A terminal window with a dark background and green text. The title bar shows the user 'iekozin@fedora' and the directory '~/work/arch-pc/lab06'. The terminal contains the following commands and their outputs:

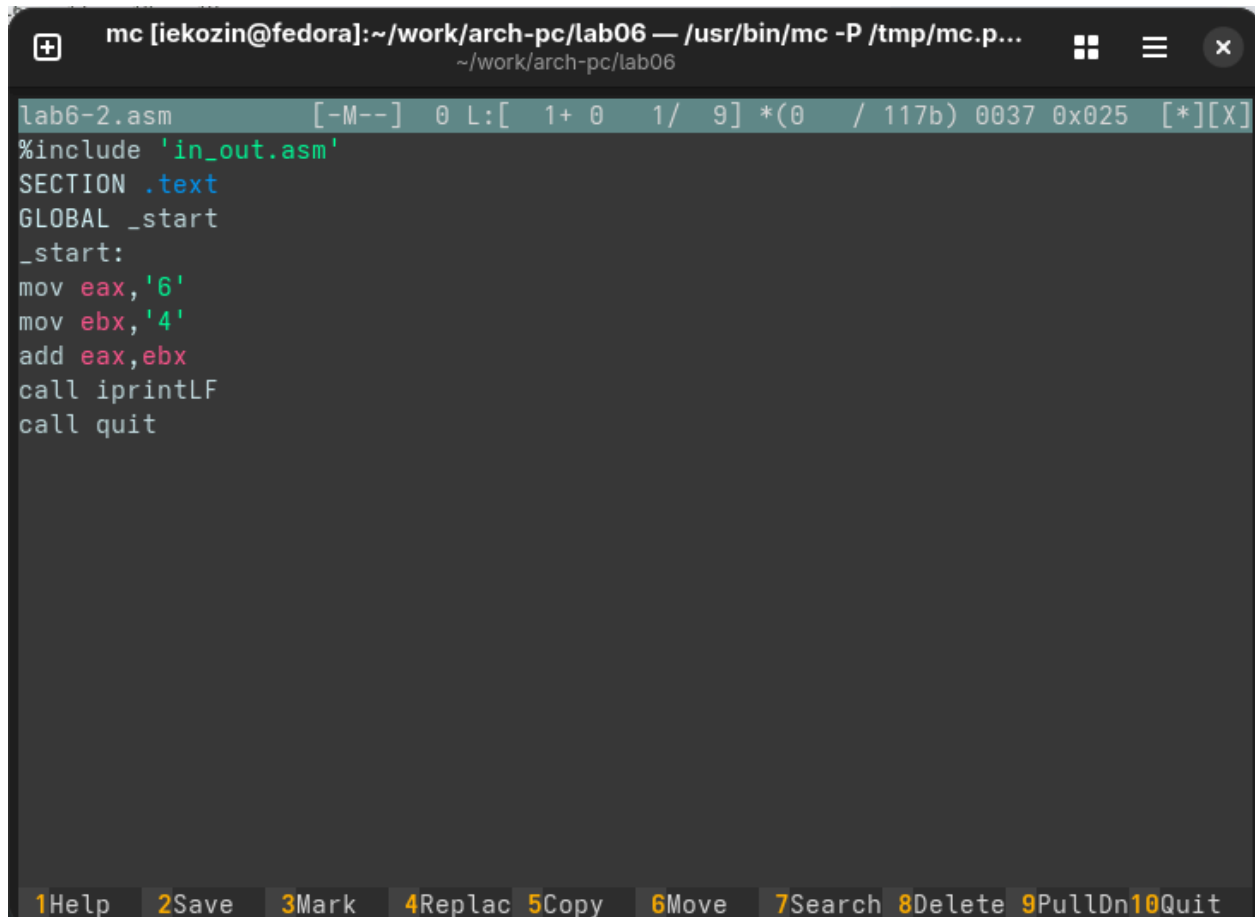
```
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
iekozin@fedora:~/work/arch-pc/lab06$ mc

iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-1

iekozin@fedora:~/work/arch-pc/lab06$
```

*Рис. 5: Запуск измененной программы*

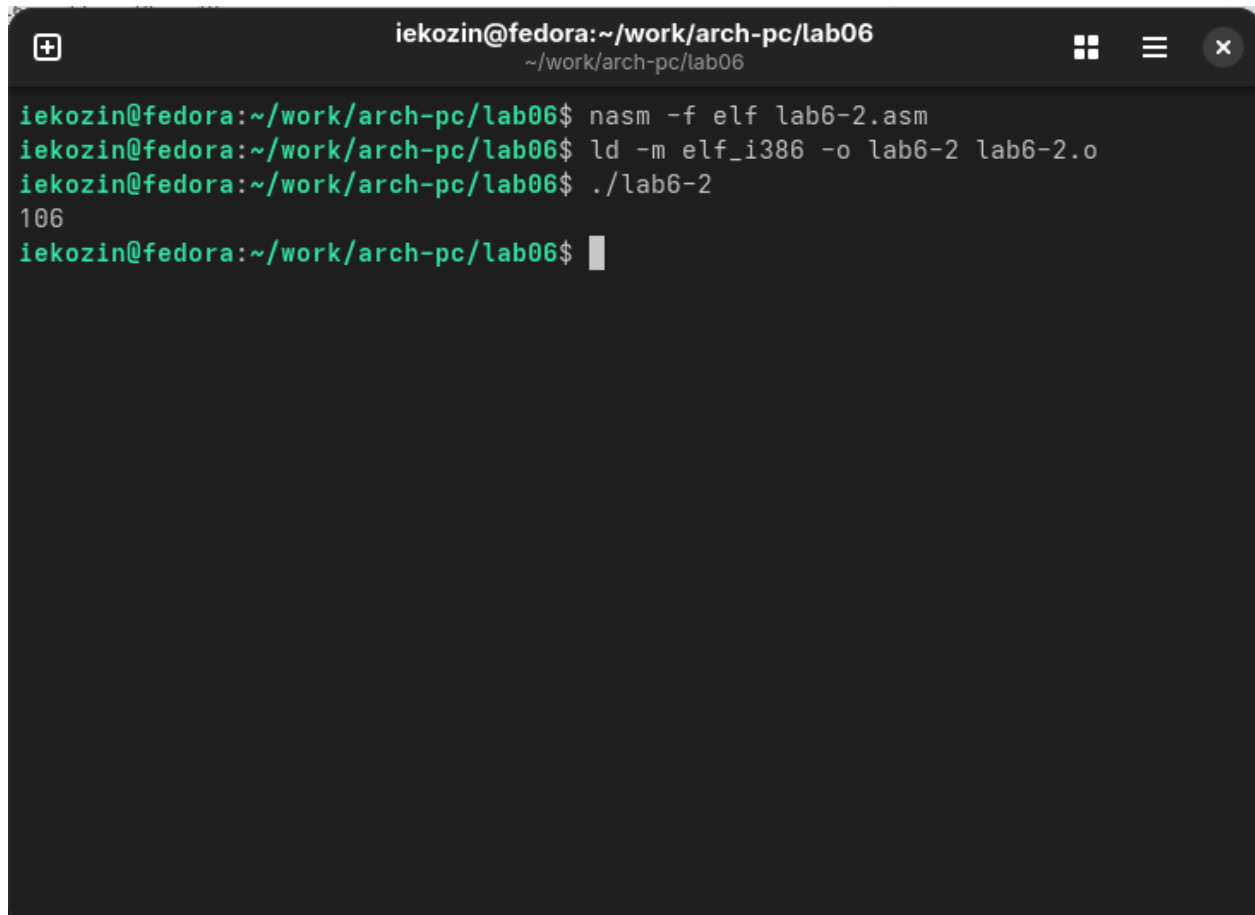
Создаю новый файл для будущей программы и записываю в нее код из листинга (рис. 6).



```
mc [iekozin@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.p...  
~ /work/arch-pc/lab06  
lab6-2.asm [-M--] 0 L:[ 1+ 0 1/ 9] *(0 / 117b) 0037 0x025 [*][X]  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax, '6'  
mov ebx, '4'  
add eax, ebx  
call iprintLF  
call quit  
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 6: Вторая программа

Создаю исполняемый файл и запускаю его, теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на iprintLF (рис. 7).

A terminal window with a dark background. The title bar shows the user 'iekozin@fedora' and the directory '~/work/arch-pc/lab06'. The terminal contains the following text:

```
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
iekozin@fedora:~/work/arch-pc/lab06$
```

*Рис. 7: Вывод второй программы*

Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат. (рис. 8).

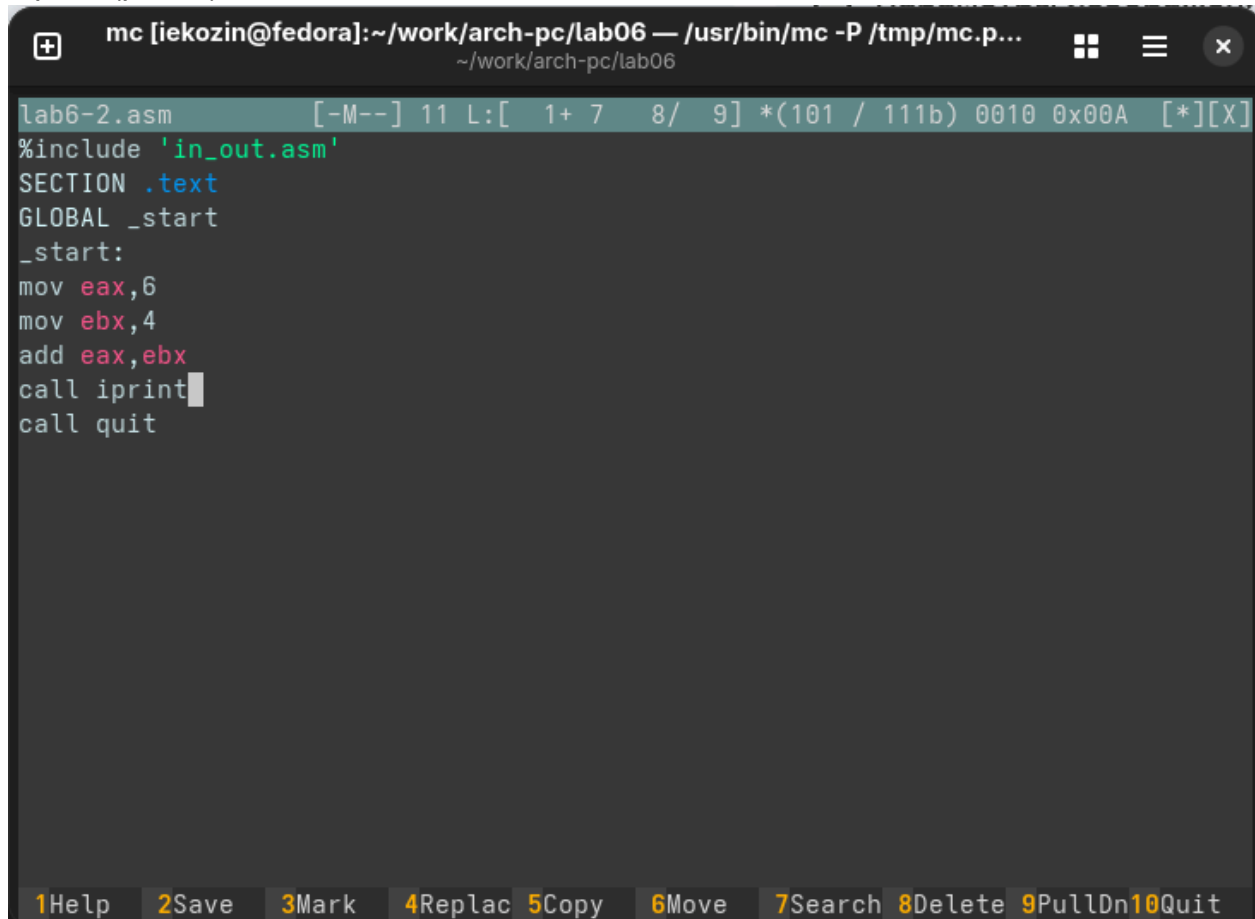


```
mc [iekozin@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.p...  
~/work/arch-pc/lab06  
lab6-2.asm [-M--] 9 L:[ 1+ 4 5/ 9] *(67 / 113b) 0010 0x00A [*][X]  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprintLF  
call quit  
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

```
iekozin@fedora:~/work/arch-pc/lab06  
~/work/arch-pc/lab06  
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm  
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-2  
106  
iekozin@fedora:~/work/arch-pc/lab06$ mc  
  
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm  
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o  
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-2  
10  
iekozin@fedora:~/work/arch-pc/lab06$
```

Рис. 8: Вывод измененной второй программы

Заменяв функцию вывода на `iprint`, я получаю тот же результат, но без переноса строки (рис. 9).



```
lab6-2.asm [-M--] 11 L:[ 1+ 7 8/ 9] *(101 / 111b) 0010 0x00A [*][X]
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit

```
iekozin@fedora:~/work/arch-pc/lab06
~/work/arch-pc/lab06

iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
iekozin@fedora:~/work/arch-pc/lab06$ mc

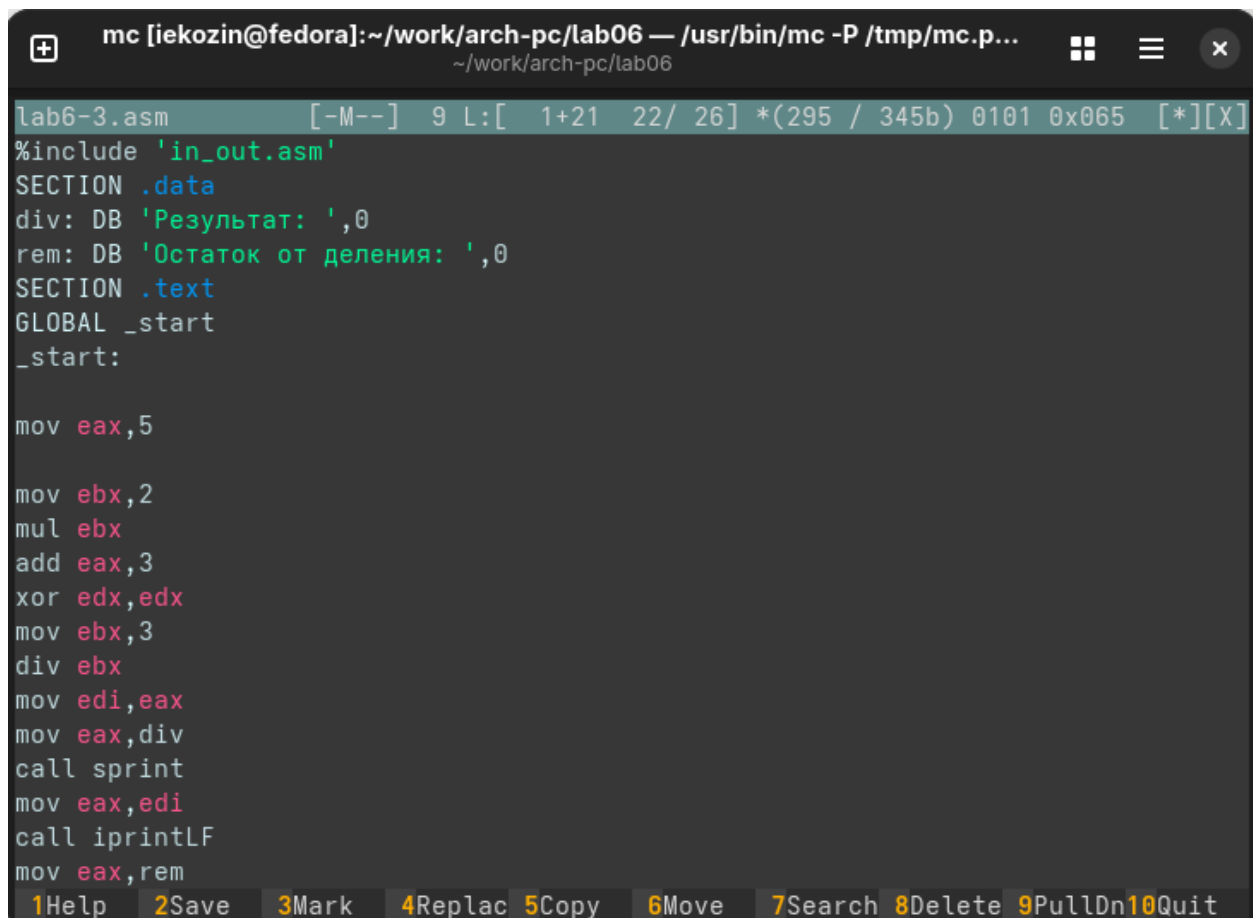
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
iekozin@fedora:~/work/arch-pc/lab06$ mc

iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-2
10iekozin@fedora:~/work/arch-pc/lab06$
```

Рис. 9: Замена функции вывода во второй программе

## 4.2 Выполнение арифметических операций в NASM

Создаю новый файл и копирую в него содержимое листинга (рис. 10).



```
mc [iekozin@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.p...
~/work/arch-pc/lab06
lab6-3.asm [-M--] 9 L:[ 1+21 22/ 26] *(295 / 345b) 0101 0x065 [*][X]
#include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:

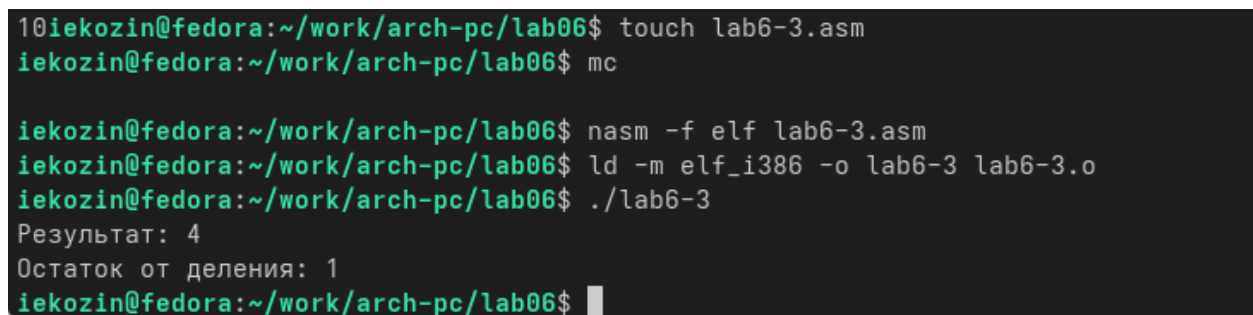
mov eax,5

mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx
mov edi,eax
mov eax,div
call sprint
mov eax,edi
call iprintLF
mov eax,rem

1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 10: Третья программа

Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. 11).

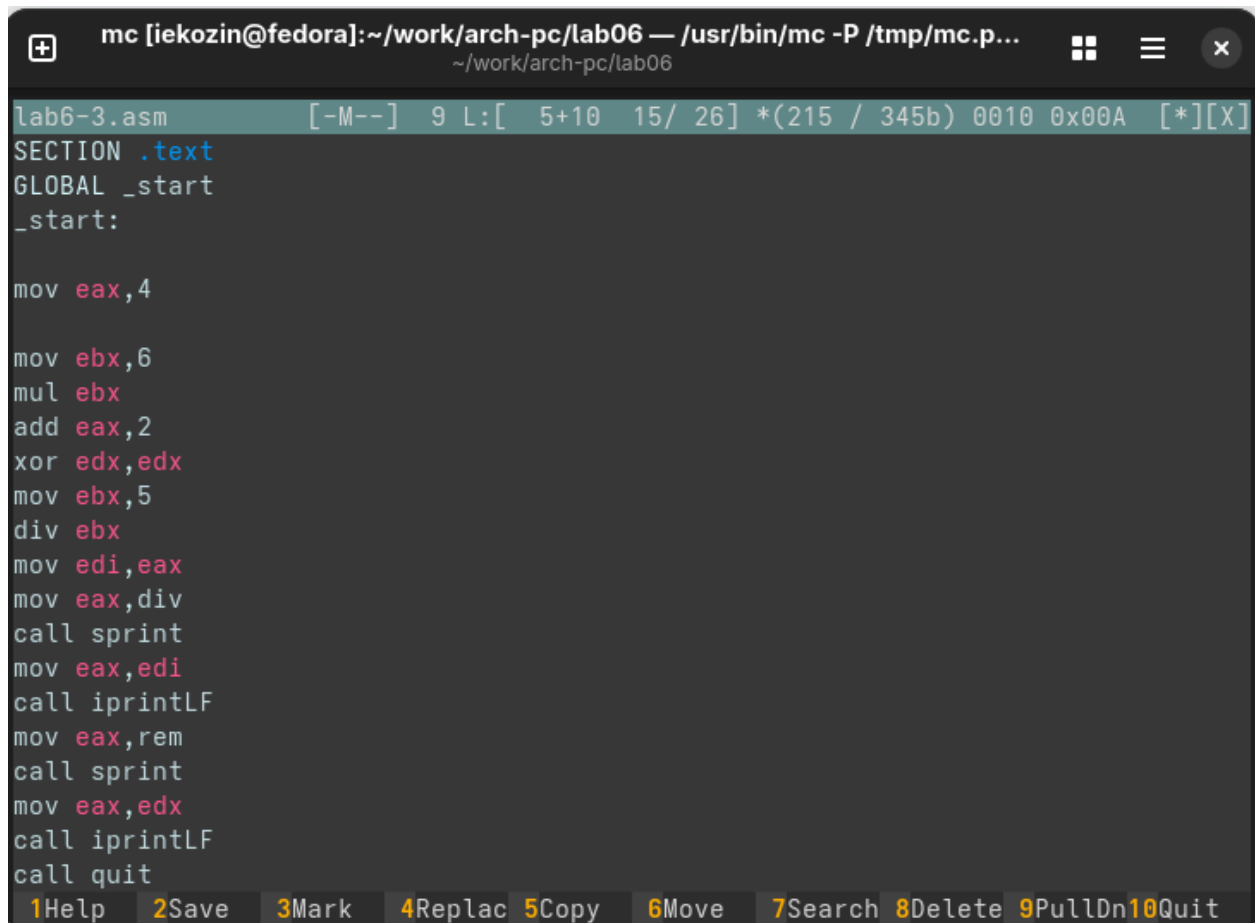


```
iekozin@fedora:~/work/arch-pc/lab06$ touch lab6-3.asm
iekozin@fedora:~/work/arch-pc/lab06$ mc

iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
iekozin@fedora:~/work/arch-pc/lab06$
```

Рис. 11: Запуск третьей программы

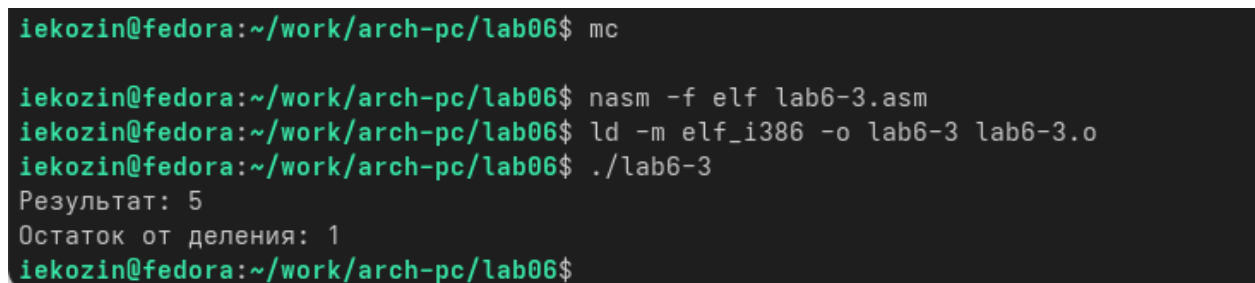
Заменяв переменные в программе для выражения  $f(x) = (4*6+2)/5$  (рис. 12).



```
mc [iekozin@fedora]:~/work/arch-pc/lab06 — /usr/bin/mc -P /tmp/mc.p...  
~ /work/arch-pc/lab06  
lab6-3.asm [-M--] 9 L:[ 5+10 15/ 26] *(215 / 345b) 0010 0x00A [*][X]  
SECTION .text  
GLOBAL _start  
_start:  
  
mov eax,4  
  
mov ebx,6  
mul ebx  
add eax,2  
xor edx,edx  
mov ebx,5  
div ebx  
mov edi,eax  
mov eax,div  
call sprint  
mov eax,edi  
call iprintLF  
mov eax,rem  
call sprint  
mov eax,edx  
call iprintLF  
call quit  
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

Рис. 12: Изменение третьей программы

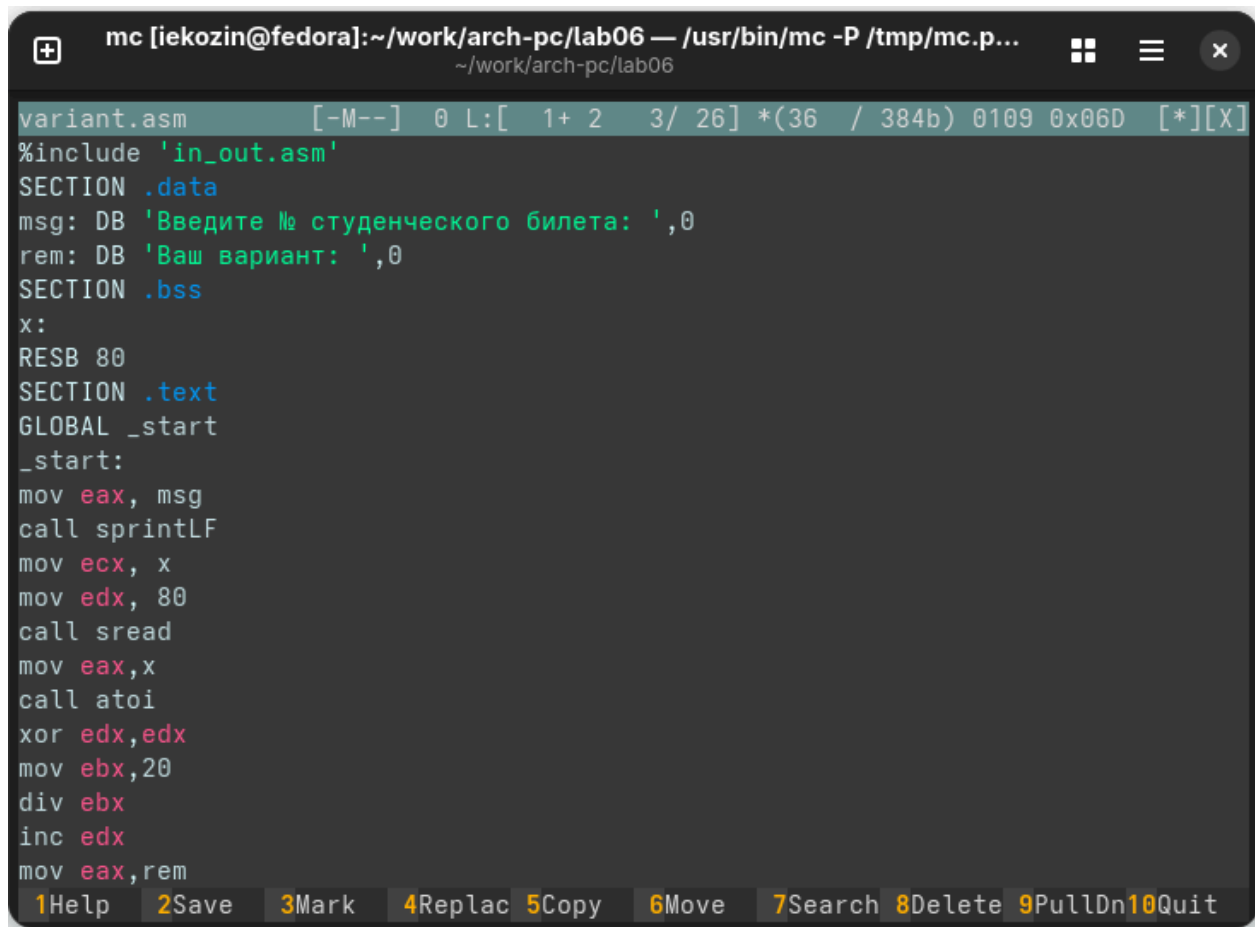
Запуск программы дает корректный результат (рис. 13).



```
iekozin@fedora:~/work/arch-pc/lab06$ mc  
  
iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm  
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o  
iekozin@fedora:~/work/arch-pc/lab06$ ./lab6-3  
Результат: 5  
Остаток от деления: 1  
iekozin@fedora:~/work/arch-pc/lab06$
```

Рис. 13: Запуск измененной третьей программы

Создаю новый файл и помещаю текст из листинга (рис. 14).



```
variant.asm [-M--] 0 L:[ 1+ 2 3/ 26] *(36 / 384b) 0109 0x06D [*][X]
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x:
RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

Рис. 14: Программа для подсчета варианта

Запустив программу и указав свой номер студенческого билета, я получил свой вариант для дальнейшей работы. (рис. 15).

```
iekozin@fedora:~/work/arch-pc/lab06
~/work/arch-pc/lab06

iekozin@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
iekozin@fedora:~/work/arch-pc/lab06$ mc

iekozin@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
iekozin@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
iekozin@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032253543
Ваш вариант: 4
iekozin@fedora:~/work/arch-pc/lab06$
```

Рис. 15: Запуск программы для подсчета варианта

### 4.3 Ответы на контрольные вопросы

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax,rem
call sprint
```

2. Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры.
3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`.
4. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
mov ebx,20 ; ebx = 20
```

```
div ebx ; eax = eax/20, edx - остаток от деления  
inc edx ; edx = edx + 1
```

5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`.
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1.
7. За вывод на экран результатов вычислений отвечают строки:

```
mov eax,edx  
call iprintLF
```

#### 4.4 Задание для самостоятельной работы

В соответствии с выбранным вариантом, я реализую программу для подсчета функции  $f(x) = 10 + (31x - 5)$ , проверка на нескольких переменных показывает корректное выполнение программы (рис. 16).



```
mc [iekozin@fedora]:~/work/study/2025-2026/Архитектура компьюте...  
~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab06/report  
lab6-4.asm [----] 15 L:[ 1+ 0 1/ 34] *(15 / 495b) 0116 0x074 [*][X]  
%include 'in_out.asm'  
  
SECTION .data  
msg: DB 'Введите значение переменной x: ',0  
rem: DB 'Результат: ',0  
  
SECTION .bss  
x: RESB 80  
  
SECTION .text  
GLOBAL _start  
  
_start:  
    mov eax, msg  
    call sprint  
    mov ecx, x  
    mov edx, 80  
    call sread  
    mov eax, x  
    call atoi  
  
    sub eax, 1  
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn10Quit
```

```
iekozin@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab06/  
report$ nasm -f elf lab6-4.asm  
iekozin@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab06/  
report$ ld -m elf_i386 -o lab6-4 lab6-4.o  
iekozin@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab06/  
report$ ./lab6-4  
Введите значение переменной x: 10  
Результат: 11Segmentation fault (core dumped)  
iekozin@fedora:~/work/study/2025-2026/Архитектура компьютера/arch-pc/labs/lab06/  
report$
```

Рис. 16: Запуск и проверка программы

Прилагаю код своей программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg: DB 'Введите значение переменной x: ',0
```

```
rem: DB 'Результат: ',0
```

```
SECTION .bss
```

```
x: RESB 80
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
    mov eax, msg
```

```
    call sprint
```

```
    mov ecx, x
```

```
    mov edx, 80
```

```
    call sread
```

```
    mov eax, x
```

```
    call atoi
```

```
    sub eax, 1
```

```
    mov ebx, 3
```

```
    mul ebx
```

```
    mov ebx, 4
```

```
    xor edx, edx
```

```
    div ebx
```

```
    add eax, 5
```

```
    mov edi, eax
```

```
    mov eax, rem
```

```
    call sprint
```

```
mov eax, edi  
call iprint
```

## 5 Выводы

При выполнении данной лабораторной работы я освоил арифметические инструкции языка ассемблера NASM.

## 6 Список литературы

1. Пример выполнения лабораторной работы
2. Курс на ТУИС
3. Лабораторная работа №6
4. Программирование на языке ассемблера NASM Столяров А. В.